Trace-Routing in 3D Wireless Sensor Networks: A Deterministic Approach with Constant Overhead

Su Xia
Cisco System Inc.
170 W Tasman Dr
San Jose, CA 95134
suxia.ull@gmail.com

Hongyi Wu
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70503
wu@cacs.louisiana.edu

Miao Jin
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70503
mjin@cacs.louisiana.edu

ABSTRACT

We propose a distributed and deterministic routing algorithm with constant storage, communication and computation overhead, dubbed *trace-routing*, for strong-connected 3D wireless sensor networks. Its basic idea is to construct a virtual cutting plane that intersects boundary surface to yield a trace, along which a routing path with guaranteed delivery can be established. We prove the correctness of trace-routing under both continuous and discrete settings. We implement the trace-routing algorithm on Crossbow sensors and carry out extensive simulations to evaluate its routing efficiency.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

Keywords

Wireless Sensor Networks; 3D; Routing

1. INTRODUCTION

This work aims to achieve *DeterminIStic routing with Constant Overhead (or DISCO routing)* in the emerging three-dimensional (3D) wireless sensor networks [1–12]. DISCO is highly desired due to the stringent resource constraints on individual nodes and the often-needed large-scale deployment of 3D wireless sensor networks. Constant overhead signifies the storage, communication and computation required for routing are bounded by a constant at each sensor node, while deterministic routing means a routing path can be determined without random search. The cost of employing DISCO routing is the potentially sub-optimal routes, which are, however, tolerable in sensor networks where data traffic is light.

The conventional table-driven routing is obviously non-DISCO, because the size of a routing table grows with the size of the network. Most on-demand routing algorithms developed for mobile ad hoc networks result in non-constant communication overhead for route discovery and thus are not DISCO either. On the other hand, while the sensor network protocols that aggregate data from sensors to sink(s) are predominantly DISCO, they do not support generic communication between any peers in the network. The earliest endeavor to achieve DISCO in large-scale peep-to-peer wireless sensor networks is geometric routing [13–20].

MobiHoc'14, August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632977.

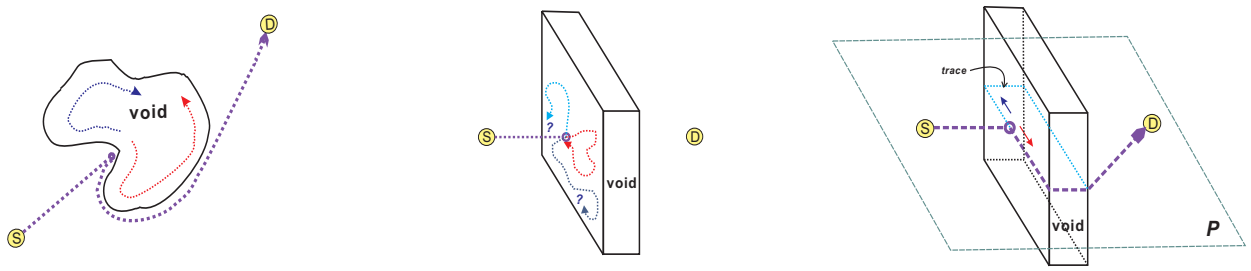
1.1 Overview of Geometric Routing

We begin our discussion on geometric routing in 2D wireless sensor networks. Given the practical limit on radio range, the communication graph of a large-scale sensor network often well preserves its geometric characteristics.

In geometric greedy routing, a node always forwards a packet to one of its neighbors closest to the destination of the packet. Geometric greedy routing is distributed with constant storage, communication and computation overhead. However, geometric greedy routing itself does not guarantee delivery of data packets, due to the presence of local minima in the network. A node is called a *local minimum* if it is not the destination but closer to the destination than all of its neighbors. Clearly, geometric greedy routing fails at local minima. As to be discussed in Sec. 3, local minima may appear at either internal or boundary nodes. The former can be resolved by using local information within two hops. Therefore, most geometric routing algorithms focus on the latter, largely following two strategies: face routing [13–20] and greedy embedding [21–27]. Face routing and its alternatives and enhancements are based on the fact that a void in a 2D planar network is a face with a simple line boundary. Thus a local deterministic algorithm can be employed to search the boundary in either clockwise or counter-clockwise direction to guide the packet out of the local minimum as illustrated in Fig. 1(a). Greedy embedding aims to embed a sensor network into a domain without local minima such that geometric routing under such embedding is always successful.

1.2 Challenges in 3D Geometric Routing

While increasing space dimension appears irrelevant to network communication protocols at the first glance, surprising challenges are observed in efforts to extend geometric routing techniques from 2D to 3D. First, none of the greedy embedding algorithms in the literature [21–27] can be generalized to 3D sensor networks. At the same time, the boundary of a void in a 3D network is no longer a closed curve or line segment, but a surface, rendering face routing infeasible. More specifically, in contrast to the simple line boundary in 2D that can be searched deterministically, there exist an arbitrarily large number of possible paths on the boundary surface to be explored in order to escape from a local minimum in 3D (as shown in Fig. 1(b)). As a matter of fact, a recent finding shows that there



(a) Face routing succeeds in 2D. (b) Face routing fails in 3D. (c) Proposed trace-routing.

Figure 1: Exposition of challenges to extending face routing from 2D to 3D and illustration of the proposed trace-routing algorithm. (a) Face routing succeeds in a 2D planar network where a local deterministic algorithm can be employed to search the boundary in clockwise or counter-clockwise direction to guide the packet out of the local minimum. (b) Face routing fails in a 3D network because there exist an arbitrarily large number of possible paths on the boundary surface to be explored in order to escape from a local minimum. (c) The basic idea of trace-routing is to establish a cutting plan that intersects the boundary surface to yield a trace, along which the packet can move out of the local minimum.

does not exist a deterministic algorithm to guarantee data delivery based on local information only in general 3D networks [28].

Given the challenges in extending 2D geometric routing algorithms to 3D, several approaches have been developed specifically for 3D settings, which fall into four categories as summarized below. The first category focuses on topology control [29], where a critical communication distance is suggested to eliminate local minima in 3D networks. However, such critical communication distance is often too large for practical applications, given the short sensor radio range, e.g., about 10 meters for MICAz motes.

The second category is based on dimension reduction by projecting nodes in 3D space onto 2D plane, where face routing is applied [6, 7]. However face routing on the projected plane does not guarantee a packet to move out of local minimum in the original 3D network. Separately, a 4D stereographic projection scheme is adopted in [9] for load balancing, but it does not address possible local minima.

The third category adopts the divide-and-conquer strategy to compress geometric information required to escape from local minima in a 3D network. To this end, tree structure or space partition is often employed. For example, a convex hull-based tree is introduced in GDSTR-3D [10]. If a local minimum is reached, the packet is forwarded according to the tree. As another example, a 3D partial unit Delaunay triangulation algorithm is proposed in [5] to divide the network space into closed subspaces such that a local minimum can be recovered within a few subspaces only. In [30], a distributed multi-dimensional tree structure is applied to support guaranteed packet delivery. Separately, the Random-Walk algorithm [8] employs a spherical dual graph structure to jump out of local minima. In all of these schemes, certain structures must be maintained by individual sensors, which are often non-locally-deterministic or requires a non-constant storage space that increases with the network size.

The fourth category is based on mapping schemes. For example, volumetric harmonic mapping is introduced in [31] to achieve deterministic and guaranteed delivery in 3D sensor networks. However, a routing table must be maintained for a network that contains more than one voids, with its size proportional to the number of voids in the network.

Moreover, all algorithms discussed in the above four categories suffer from network dynamics. For example, when sensor nodes move, they must frequently update their data structures or virtual coordinates. Such updates are usually expensive or even unattainable.

1.3 Contributions of This Work

As discussed above, none of the existing schemes support DISCO routing in 3D wireless sensor networks. In this research, we consider a sensor network deployed in a 3D space with one or multiple internal holes, where sensors cannot be deployed. We show that local minima are always on the boundaries of holes if nodal density is high, and there exists a DISCO algorithm to support routing between any pair of points in a strong-connected 3D network. We propose a distributed and deterministic algorithm with constant storage, communication and computation overhead, dubbed trace-routing, to escape from local minima. Trace-routing is triggered when a packet reaches a local minimum. It constructs a virtual cutting plane that contains the local minimum and the destination and intersects the corresponding boundary surface to yield a trace. The trace is a closed loop that can be computed locally with constant overhead. The packet is routed along such a trace, thus deterministically moving out of the local minimum as illustrated in Fig. 1(c).

The proposed trace-routing algorithm does not demand preprocessing of the global network information, neither does it require establishing or maintaining a global data structure, which often consumes a storage space proportional to the network size and needs frequent update due to network dynamics. We further prove the trace-routing algorithm guarantees data delivery in a strong-connected 3D network under both continuous and discrete settings. In a nutshell, trace-routing supports DISCO, thus highly efficient in large-scale 3D sensor networks. It is worth mentioning that the proposed trace-routing algorithm and related discussions and proofs do not rely on any particular communication model (such as the unit ball graph model or quasi-unit ball graph model). Only a maximum radio range is assumed, which is generally known in practical sensor networks.

The rest of this paper is organized as follows: Sec. 2 introduces the proposed trace-routing algorithm in continuous 3D domain. Sec. 3 presents a practical design of trace-routing and proves its correctness under discrete 3D sensor network settings. Secs. 4 and 5 discuss simulation and experimental results. Finally, Sec. 6 concludes the paper.

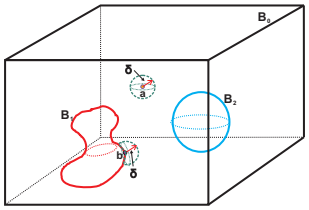


Figure 2: Boundaries and δ-vicinities.

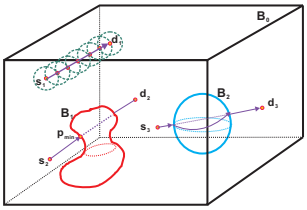


Figure 3: Geometric greedy routing.

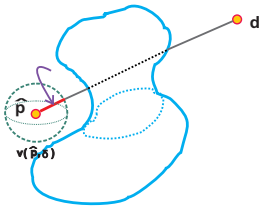


Figure 4: Illustration of Lemma 2.

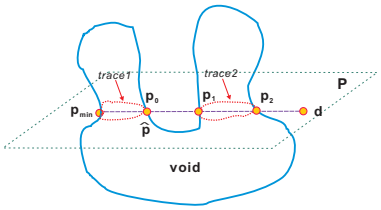


Figure 5: Illustration of Lemma 4.

2. TRACE-ROUTING IN CONTINUOUS 3D SPACE

As shown in [28], there does not exist a universal deterministic algorithm to guarantee data delivery based on local information only in general 3D networks. Following this result, a natural question is whether there exists a DISCO routing algorithm for a class of practical 3D networks and how to develop a distributed algorithm to achieve such DISCO routing. To deliver a lucid conceptual presentation and gain useful theoretic insights, we first investigate the failures in geometric greedy routing and present our proposed algorithm in a continuous 3D space. We will discuss the algorithm under discrete 3D sensor network settings in the next section.

2.1 Greedy Routing in Continuous 3D Volume

Let's start with several definitions that are necessary to our exposition and discuss the essence of geometric greedy routing in a continuous 3D Euclidean space. Let $\mathbf{U}$ denote an Euclidean volume in 3D space $\mathbb{R}^3$, which is enclosed by a set of boundaries $\mathbf{B} = \{B_k \mid 0 \leq k \leq K\}$, where B_0 is the outer boundary and B_k $(k > 0)$ is an inner boundary of $\mathbf{U}$. Let $L(p, q)$ denote a straight line segment from Point p to Point q, and $|L(p, q)|$ denote its Euclidean distance. Let Λ_k denote the area enclosed by B_k $(k \geq 0)$. Obviously, $\mathbf{U} = \Lambda_0 \setminus \cup\Lambda_k$ $(1 \leq k \leq K)$.

DEFINITION 1. *Let $C(s, d)$ be a sequence of line segments, $C(s, d) = \langle L(p_0, p_1), L(p_1, p_2), .., L(p_{m-1}, p_m)\rangle$, where $p_0 = s$ and $p_m = d$. We call $C(s, d)$ a routing path connecting s and d, if and only if*

- *$L(p_i, p_{i+1})$ does not intersect with $L(p_j, p_{j+1})$, $\forall\ i \neq j$, $i + 1 \neq j$ and $i \neq j + 1$; and*
- *$L(p_i, p_{i+1})$ does not penetrate any boundary of $\mathbf{U}$, $\forall\ L(p_i, p_{i+1}) \in C(s, d)$.*

The first condition of Definition 1 requires the routing path free of self-intersection, while the second condition indicates that the path must be completely contained inside $\mathbf{U}$. Next we introduce the geometric greedy routing algorithm via the following two definitions.

DEFINITION 2. *We call $V(p, \delta)$ the δ-vicinity of Point p, if $\forall\ \hat{p}$ in $V(p, \delta)$, $L(p, \hat{p})$ is completely contained in $\mathbf{U}$ and $|L(p, \hat{p})| \leq \delta$ where δ is a given positive real number.*

Fig. 2 illustrates two examples of δ-vicinity of points at different locations. The δ-vicinity of Point a, an internal node, is a ball centered at a with a radius of δ, while Point b has a defected δ-vicinity due to the nearby boundary.

DEFINITION 3. *A routing path $C(s, d)$ is a geometric greedy path (or greedy path for conciseness), denoted as $C(s, d, \delta)$, if $\forall$ p_i on $C(s, d, \delta)$, p_{i+1} is the closest point in $V(p_i, \delta)$ to Destination d.*

The above definition suggests a simple geometric greedy routing algorithm. It starts from the source as its current point, and chooses the point in its current δ-vicinity that is the closest to the destination as the next point. The process repeats until it reaches the destination. An example of routing from Source s_1 to Destination d_1 is shown in Fig. 3. Note that the geometric greedy routing algorithm may fail to find the next point, when the current point is closer to the destination than all other points in its δ-vicinity. Formally we define the point where geometric greedy routing fails as a local minimum.

DEFINITION 4. *Point p_{min} is a local minimum for Destination d under a given δ, if $p_{min} \neq d$ and $L(p_{min}, d) \leq L(p, d)\ \forall\ p \in V(p_{min}, \delta)$.*

An example of geometric greedy routing failure is illustrated in Fig. 3 from Source s_2 to Destination d_2. Greedy routing often fails when the routing path meets the boundary of an internal hole. Particularly, we have the following observation regarding geometric greedy routing failures.

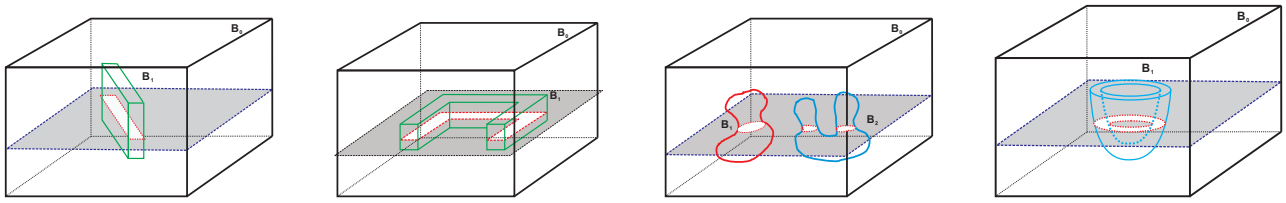
LEMMA 1. *If there does not exist a geometric greedy routing path $C(s, d, \delta)$ between s and d under a given δ, $L(s, d)$ must intersect at least one of the boundaries of $\mathbf{U}$.*

PROOF. We prove it by contradiction. Assume $L(s, d)$ does not intersect any boundary. Since s and d are points in $\mathbf{U}$, $L(s, d)$ must be completely contained in $\mathbf{U}$ (otherwise it would intersect a boundary). Then a greedy path $C(s, d, \delta)$ along the straight line $L(s, d)$ can be readily constructed according to Definition 3, contradicting the given condition that there is no greedy path between s and d. Thus, the lemma is proven. □

Equivalent to Lemma 1, if $L(s, d)$ does not intersect any boundary of $\mathbf{U}$, we can always find a greedy routing path from s to d. It gives the sufficient condition for the existence of a greedy path. But it is not the necessary condition, i.e., even if $L(s, d)$ intersects a boundary, a greedy routing path may still be established from s to d. Fig. 3 illustrates such an example where $L(s_3, d_3)$ intersects a spherical boundary, but there exists a greedy routing path along the spherical boundary to Destination d_3. Based on Lemma 1 and Definition 4, we arrive at the following conclusion.

LEMMA 2. *Greedy routing only fails at local minimums and local minimums are always on the boundaries of holes.*

PROOF. The first part of the lemma is obvious from Definitions 3 and 4. The second part of the lemma can be proven by contradiction. Assume there is a local minimum point $\hat{p}$ that is an internal point in $\mathbf{U}$ (i.e., not on any boundary). Let's draw a line $L(\hat{p}, d)$. Since $\hat{p}$ is an internal point, there must exist a δ such that the intersection of $L(\hat{p}, d)$ and $V(\hat{p}, \delta)$ (highlighted in red in Fig. 4) is a line segment, on which all points are closer to d than $\hat{p}$ is. This contradicts the assumption that $\hat{p}$ is a local minimum. The lemma is proven. □



(a) Volumes with convex boundaries are always s-con.

(b) A volume with a C-shape inner void is s-con.

(c) A volume with complex shapes of inner voids is s-con.

(d) A volume with a bowl-shape inner void is not s-con.

Figure 6: Examples of s-con and non-s-con volumes. Each volume has a cubic outer boundary (i.e., B_0) and one or two inner voids delineated by the colored boundaries. An example of cutting planes is shown in each figure, where the shaded area(s) illustrate its intersection with the volume.

Lemma 2 reveals an important fact: greedy routing only fails at boundaries, inspiring us to develop a deterministic boundary navigation algorithm to escape from local minimums.

2.2 Trace-Routing in Continuous 3D Volume

In a 3D space for practical sensor network deployment, the boundary surfaces are *closed* (a closed surface is a compact one with empty boundary) and the 3D volume is often *Strong-CONnected (s-con)*. Here "*closed*" means no "breaches" on the surface. The s-con property is defined below.

DEFINITION 5. *A 3D volume **U** is s-con (*Strong-CON-nected*), if and only if the intersection of* any *plane and **U** is a connected graph on the plane.*

Figs. 6(a)-6(c) depict several examples of s-con 3D volumes. Each volume has a cubic outer boundary (i.e., B_0) and one or two inner voids delineated by the colored boundaries. An example of cutting planes is shown in each figure, where the shaded area(s) illustrate its intersection with the volume. It is obvious that a 3D volume with convex boundaries is always s-con as shown in Fig. 6(a). But s-con volumes may have non-convex boundaries too. For instance, volumes in Fig. 6(b) and Fig. 6(c) are s-con with non-convex inner boundaries. Note that, although the plane intersects the volume in Fig. 6(c) to yield multiple voids on the intersection plane, the intersection itself (i.e., the shaded area) is still connected. Therefore it is s-con. Fig. 6(d) shows an example of non-s-con volume, with a bowl shape inner void, resulting in disconnected components on the intersection plane (see the disconnected shaded areas).

Although DISCO routing is not achievable in general 3D networks as discussed earlier, we now introduce the proposed trace-routing algorithm, and prove that it supports DISCO routing in any 3D wireless sensor networks that are Strong-CONnected (s-con). There exist other types of boundary surfaces in theory, but they are rare under practical sensor network settings and thus excluded from our discussions below.

As shown by Lemma 2, if greedy routing fails, it must stuck at a local minimum, and such local minimum must be on a boundary of **U**. Let p_{min} denote the local minimum where geometric greedy routing fails, and assume p_{min} is on Boundary B_i. Now we construct an arbitrary plane that contains $L(p_{min}, d)$. It intersects B_i, resulting in one or multiple traces. By examining the traces, we have the following observation.

LEMMA 3. *A trace on a boundary surface is a closed loop with no self-intersection.*

PROOF. If a trace is not a closed loop, the space between two endpoints on the plane forms a *breach* on the surface, contradicting the fact of closed boundary surface. Since the trace is the intersection of the surface and the plane, it cannot self-intersect, given that the surface is homeomorphic to sphere with no self-intersecting component. □

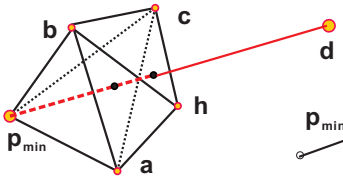


Figure 7: Illustration of Lemma 5.

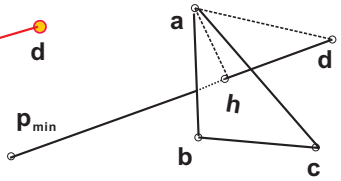


Figure 8: Prove of Theorem 2.

If we draw a line segment from p_{min} to d, $L(p_{min}, d)$ must intersect B_i at two or more points including p_{min}, as shown in Fig. 5. Let p_0 denote one of such intersection points, which is the closest to p_{min}. Clearly, p_0 must be closer to d than p_{min} is, since it is on the line segment between p_{min} and d.

LEMMA 4. *In an s-con volume, the trace containing p_{min} must also contain p_0 and there is a loop-free path from p_{min} to p_0 along this trace.*

PROOF. Obviously p_{min} and p_0 must belong to some trace(s). According to the way of constructing the plane, $L(p_{min}, d)$ must intersect one or multiple traces, yielding a sequence of intersection points on $L(p_{min}, d)$. Now let us look at the trace that contains p_{min}, denoted by $\Gamma_{p_{min}}$. Since $L(p_{min}, d)$ starts from p_{min} that is on Boundary B_i and goes toward the inside of B_i, it cannot reach another point before it meets $\Gamma_{p_{min}}$ again. Let $\hat{p}$ denote this intersection point as shown in Fig. 5. Since $\hat{p}$ is the first such intersection point, it is the closest to p_{min}. Clearly, $\hat{p}$ must be co-located with p_0 introduced earlier. Therefore, we have proven p_{min} and p_0 must be on the same trace.

As shown by Lemma 3, a trace is a closed loop with no self-intersection. Thus it is straightforward to choose a loop-free route along the trace in either clockwise or counterclockwise direction to route from p_{min} to p_0. □

Till now we are ready to formally introduce the trace-routing algorithm and prove its correctness as follows.

THEOREM 1. *There exists a deterministic algorithm with constant storage, communication and computation overhead, which can always successfully navigate the routing path out of local minimums in an s-con volume.*

PROOF. First we construct a simple algorithm, dubbed *trace-routing* as follows. When geometric greedy routing reaches a local minimum p_{min} on Boundary B_i, it chooses a *cutting plane* that is determined by p_{min}, Destination d, and another random point $\hat{p}$. The plane intersects B_i, yielding a trace that contains p_{min}. The routing path advances along the trace in clockwise or counterclockwise direction until it reaches a point that is closer to Destination d than p_{min} is. Then geometric greedy routing follows.

Now we prove the correctness of the algorithm. According to Lemma 4, there must exist at least one point (i.e., p_0) on the trace containing p_{min}, which is closer to Destination d than p_{min} is. Since the trace is a closed loop with no self-intersection, if one starts from p_{min} and navigates along the trace in clockwise or counterclockwise direction, it will visit all points on the trace exactly once before it returns back to p_{min}. Therefore the routing path must reach p_0, from which it can escape from the local minimum.

Trace-routing is clearly deterministic as evident by the description of the algorithm itself. Next, we show its storage, communication and computation overhead are all constant. The algorithm requires each node to maintain the coordinates of itself and its neighbors only, and to pass merely a constant amount of information (i.e., the cutting plane defined by three points p_{min}, $\hat{p}$, and d) along the routing path. Assume the routing path reaches Point p. The trace segment around p can be found by computing the intersection of the cutting plane and the boundary surface in $V(p, \delta)$. Thus, the routing decision is made locally with a constant computation complexity.

In summary we have proved the existence of a local deterministic algorithm with constant storage, communication and computation overhead that can always successfully move out of local minimums in an s-con volume. □

Theorem 1 reveals salient properties of trace-routing, which supports efficient and distributed implementation under practical sensor network settings to be discussed next.

3. TRACE-ROUTING IN DISCRETE 3D NETWORKS

While trace-routing has been introduced above to escape from local minimums, it remains challenging to adapt the concepts and ideas to a practical sensor field. In contrast to the continuous 3D Euclidean volume considered in Sec. 2, a sensor network is under a discrete setting, which presents an approximation of the 3D volume only, rendering part of the earlier discussed methods and results invalid. For example, sensor nodes rarely reside perfectly on the trace computed in a continuous space, calling for approximated solutions. Furthermore, the routing algorithm must be distributed in sensor networks. However, the deployment of sensor nodes exhibits randomness and their radio transmissions are often irregular, which together lead to new challenges to realize distributed trace-routing and to achieve constant storage, communication and computation overhead.

In this section, we examine our conclusions presented in the previous section under a discrete setting, and introduce a practical design of the trace-routing algorithm for 3D sensor networks with internal holes. The following discussions do not rely on any particular communication model (such as the unit ball graph model or quasi-unit ball graph model). Only a maximum transmission range is assumed, which is generally known in practical wireless sensor networks.

3.1 Challenges of Trace-Routing in 3D Sensor Networks

Let us first revisit the concepts and conclusions given in Sec. 2 for continuous 3D Euclidean volume, and evaluate their validity in 3D sensor networks.

3.1.1 Tetrahedron Structure

To facilitate our exposition, we first establish a tetrahedral structure based on discrete sensors as discussed in [31]. A triangular face in a tetrahedral structure is a boundary face if it is associated with one and only one tetrahedron. A set of connected boundary faces form a closed triangular boundary surface. In the following discussions, a "boundary" refers to a triangular boundary surface.

3.1.2 Local Minimums

Lemma 2 has shown that local minimums are always on the boundaries of holes in a continuous 3D Euclidean volume. Unfortunately, this result no long holds in discrete settings, where local minimums may appear at not only boundary but also internal sensors. However, the internal local minimums can be resolved by using local information only as revealed by the following lemma.

LEMMA 5. *Given an internal local minimum p_{min}, the routing path can move out the local minimum by using local information in 2δ-vicinity of p_{min}, where δ is the maximum radio transmission range.*

PROOF. Based on the tetrahedron structure, a greedy routing path can be established according to faces [31]. More specifically, when the routing path reaches an internal local minimum p_{min}, a line segment from p_{min} to Destination d can be computed by p_{min}. The line segment passes through a sequence of faces in the tetrahedron structure. Let's consider the first such face, denoted by $Face(a, b, c)$, where a, b, and c are vertex nodes (as illustrated in Fig. 7). Obviously, $Face(a, b, c)$ must be closer to the destination compared to the current face, i.e., $Face(p_{min}, a, b)$. Clearly, $Face(a, b, c)$ is within two-hop neighborhood of p_{min}. As a result, the routing path can move out of the local minimum by using local information in 2δ-vicinity of p_{min}. □

According to Lemma 5, we slightly modify the greedy routing algorithm to let each node check its two-hop neighborhood to identify the closest node to the destination, thus eliminating internal local minimums. Note that, the modified algorithm does not help to escape from boundary local minimums, which are yet to be addressed by the proposed trace-routing.

3.1.3 Traces

As discussed in Sec. 2.2, a trace on an inner boundary is a closed loop without self-intersection. It can be computed by using local information. But in a discrete setting, sensors rarely reside perfectly on the computed trace. To this end, we redefine traces under discrete settings as follows.

DEFINITION 6. *In a discrete 3D space, a trace formed by a given plane and a boundary consists of a sequence of line segments on the boundary surface, which intersect the plane and are connected end-to-end to form a loop.*

LEMMA 6. *Given a tetrahedral structure of the 3D sensor network and a plane that intersects a triangular boundary surface of the tetrahedral structure, a closed-loop trace can be constructed deterministically.*

PROOF. For a given tetrahedral structure of the 3D sensor network, the faces on the boundary of an internal hole form a closed and connected triangular surface. A plane intersects the triangular boundary surface at a set of triangular faces, denoted by Φ. For each face $f \in \Phi$, it must have at least two edges intersecting the plane. Since each edge is shared by two faces on the triangular boundary surface, Face f must have at least two neighboring faces in Φ. Therefore, the faces in Φ are connected to form a closed strap. It is thus straightforward to select a sequence of edges on such faces, which intersect the plane, to build a closed loop, i.e. the trace. □

Based on Lemma 6, we have the following result that shows the correctness of trace-routing.

THEOREM 2. *A deterministic routing path can be identified by following the* trace *constructed according to Lemma 6 to escape from local minimums.*

PROOF. To prove this theorem, we show that there exists at least one node on the trace, which is closer to the destination than the local minimum (i.e., p_{min}) is. Similar to our discussion under the continuous setting, if we draw a line segment from p_{min} to Destination d, $L(p_{min}, d)$ must intersect the boundary surface at one or more points besides p_{min}. Let h denote one of such intersection points that is the closest to p_{min}, and assume h is on an arbitrary face, e.g., $Face(a, b, c)$ as illustrated in Fig. 8. Clearly, $L(a, b)$, $L(b, c)$ and $L(a, c)$ are no greater than the maximum radio range r because they are connected in the communication graph. Since h is on $Face(a, b, c)$, $L(a, h)$, $L(b, h)$ and $L(c, h)$ must be no greater than r either. At the same time, $L(p_{min}, a)$, $L(p_{min}, b)$, $L(p_{min}, c)$ and $L(p_{min}, h)$ are all greater than r, since there is a hole between p_{min} and $Face(a, b, c)$. Thus we have $L(a, d) < L(a, h) + L(h, d) \leq r + L(h, d) < L(p_{min}, h) + L(h, d) = L(p_{min}, d)$. Therefore a is closer to d than p_{min} is. Similarly, b and c are closer to d than p_{min} is. Note that at least one edge of $Face(a, b, c)$ must be on the trace constructed according to Lemma 6. Therefore one can always find a node on the trace closer to d than p_{min} is. The theorem is thus proven. □

Theorem 2 shows trace-routing is deterministic and always succeeds in discrete sensor networks. Moreover, similar to the proof of Theorem 1, its storage, communication and computation overhead are all bounded by a constant.

3.2 Proposed Trace-Routing Algorithm in 3D Sensor Networks

The basic idea of trace-routing is to move clockwise or counterclockwise around the trace constructed by Lemma 6, which is a closed loop and thus can navigate out of local minimums. The discussions so far have assumed known boundary nodes and triangular boundary surfaces. Next we show that such information can be acquired locally with constant overhead or is unnecessary at all in practice.

3.2.1 Boundary Awareness

We adopt the boundary detection algorithm proposed in [32], which is localized, requiring information within one-hop neighborhood only for each node to determine if it is on a boundary. Therefore, when greedy routing fails at a local minimum p_{min}, the boundary detection algorithm [32] is employed to identify boundary nodes in the δ-vicinity of p_{min} as input for the trace-routing algorithm.

3.2.2 Triangular Boundary Surface

The tetrahedron structure and triangular boundary surfaces are assumed to facilitate our discussions in Sec. 3.1, especially the proof of Lemmas 5 and 6. However, it is unnecessary to explicitly construct them for the implementation of trace-routing. The algorithm can simply choose one of the boundary nodes in its neighborhood as the next hop along the trace. More specifically, the trace-routing algorithm is outlined in Algorithm 1. It is initiated when a greedy routing path reaches a local minimum p_{min}. It defines a plane based on p_{min}, d and an arbitrary node p in the δ-vicinity of p_{min}. Let Δ denote the plane. The algorithm then enters an iterative process. In each iteration, a boundary node in the δ-vicinity of the current node (i.e., p_{cur}) is identified to be the next node (i.e., $\hat{p}$) along the trace. To keep the next node as close to the ideal trace (computed under continuous setting) as possible and to advance along the trace as fast as possible, $\hat{p}$ is chosen such that $L(p_{cur}, \hat{p})$ intersects Plane Δ and forms the largest angle with $L(p_{pre}, p_{cur})$, where p_{pre} is the previous hop node on the routing path. The iterative process repeats until it reaches a node closer to the destination than p_{min} is, i.e., $|L(p_{cur}, d)| < |L(p_{min}, d)|$.

It is clear that Algorithm 1 only results in constant storage, communication and computation overhead. It does not demand preprocessing of the global network information, neither does it require establishing or maintaining a global data structure, which often consumes a storage space proportional to the network size and needs frequent update due to network dynamics. Thus it is highly efficient for large-scale 3D sensor networks.

Algorithm 1: Trace-Routing Algorithm

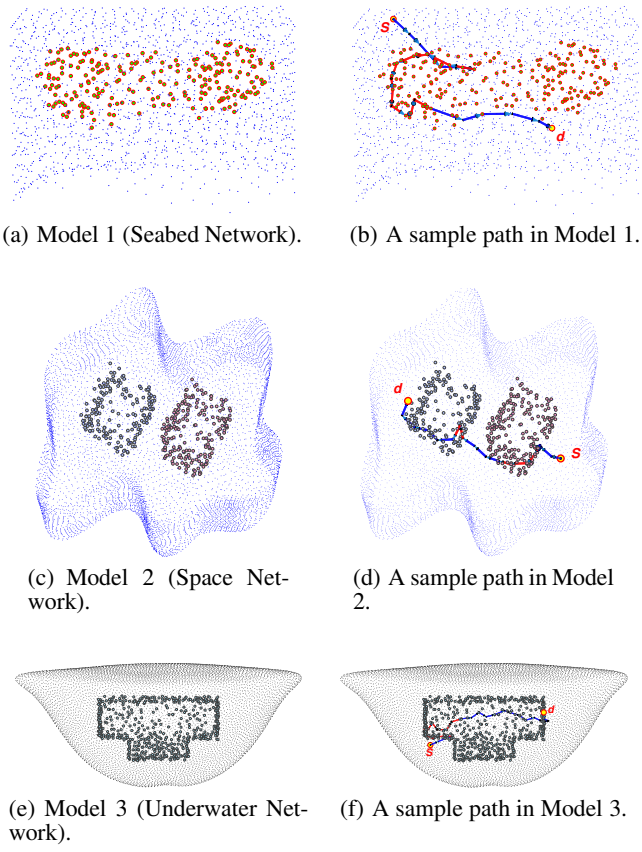
Input: Local minimum p_{min}, Destination d;
1 Define Plane Δ based on p_{min}, d and an arbitrary node $p \in V(p_{min}, \delta)$;
2 $p_{cur} \leftarrow p$, $p_{pre} \leftarrow p_{min}$;
3 **while** $|L(p_{cur}, d)| \geq |L(p_{min}, d)$ **do**
4 Identify a boundary node $\hat{p} \in V(p_{cur}, \delta)$ such that $L(\hat{p}, p_{cur})$ intersects Plane Δ and forms the largest angle with $L(p_{pre}, p_{cur})$;
5 $p_{pre} \leftarrow p_{cur}$; $p_{cur} \leftarrow \hat{p}$;

3.3 Peer-to-Peer Routing in 3D Wireless Sensor Networks

In summary, when a packet is routed from Source s to Destination d, it first follows geometric greedy routing until a local minimum is reached. Then it employs trace-routing to move out of the local minimum. When the packet arrives at a non-local-minimum node, it switches to greedy routing again. The process repeats until the packet arrives at Destination d. Examples of the routing paths are illustrated in Fig. 1(c) and Figs. 9.

4. SIMULATION RESULTS

We have implemented the proposed trace-routing algorithm (denoted by "TR") and two other state-of-the-art 3D wireless sensor network routing algorithms, namely GDSTR-3D [10] and HVE (harmonic volumetric embedding) [31], for performance evaluation and comparison. Since HVE works in networks with one hole only, we focus on the comparison between TR and GDSTR-3D in most simulation scenarios. Various 3D sensor networks with different sizes (ranging from 1,800 to 11,000 nodes) and shapes are simulated. Fig. 9 illustrates three examples, where the inner boundary



(a) Model 1 (Seabed Network). (b) A sample path in Model 1.

(c) Model 2 (Space Network). (d) A sample path in Model 2.

(e) Model 3 (Underwater Network). (f) A sample path in Model 3.

Figure 9: The left column shows 3D sensor network models and the right column illustrates sample routing paths in each model. The sensors on the boundary of holes are highlighted in different colors. The blue segments indicate greedy forwarding, while the red segments follow trace-routing.

nodes are highlighted to delineate the holes in the networks. Sensors are randomly distributed. While our proposed scheme does not rely on a particularly communication model, we carry out simulations based on unit ball graph and quasi-unit ball graph models, with an average nodal degree between 10 to 18. Our simulations show that the proposed scheme works well for networks with a nodal degree greater than 7 (which represents a sparse network in three dimensional space).

4.1 Stretch Factor

As discussed in previous sections, TR, GDSTR-3D and HVE all ensure successful delivery. Therefore we focus on stretch factor in performance evaluation. The stretch factor of a route is the ratio of the actual routing path length to the shortest path length. 20,000 pairs of nodes are randomly selected to calculate the average stretch factor for each network model. The following table compares the average stretch factors of GDSTR-3D and TR in different networks. The results of HVE are not included because it can be applied for networks with one hole only, where its stretch factor is around 1.52. Both algorithms achieve good and comparable stretch factor (less than 1.3) in all network models. GDSTR-3D is more sensitive to the hole size and the number of holes, exhibiting higher stretch factors under Models 2-4. This is mainly because that the GDSTR-3D uses two trees rooted at two furthest away nodes. When there are multiple holes or complex shape holes, the

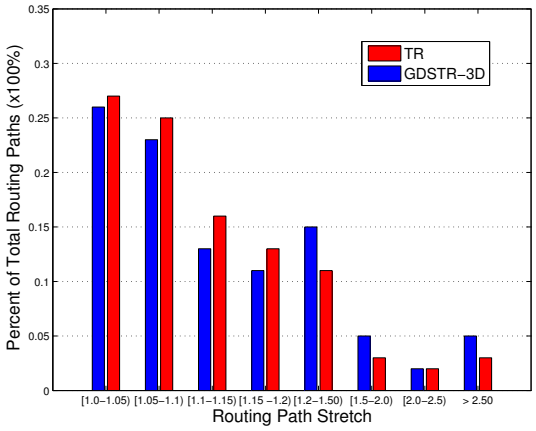


Figure 10: Stretch under UDG.

local minimum recovery procedure that follows one of the two trees often results in significantly stretched routing paths in comparison with corresponding shortest paths. Moreover, under complex and multiple hole settings, there are more overlapped convex hull partitions, which cause longer paths during the tree routing phase. In a contrast, TR follows the geometric boundary of the holes in local minimum recovery, resulting in a stable stretch factor in average.

Models	1	2	3	4	5	6	Ave.
TR	1.05	1.09	1.04	1.21	1.08	1.11	1.10
GDSTR-3D	1.14	1.23	1.29	1.25	1.07	1.19	1.20

The distribution of stretch factor is illustrated in Fig. 10. As can be seen, more than 50% routing paths have their stretch factor lower than 1.15 under both schemes. Compared with TR, the stretch factor distribution of GDSTR-3D has a noticeable shift to the right side with more routes experiencing a stretch factor of 1.2 or higher.

As discussed in the previous sections, TR and GDSTR-3D do not rely on any particular transmission model. Here we test them under a general "QUASI-UBG" model, where two nodes are connected if their distance is less than d; or disconnected if their distance is beyond the maximum transmission range r; or have a probability of p to be connected if their distance is between d and r. We vary d and p, and compare the performance of TR and GDSTR-3D. Both schemes support perfect data delivery rate. Their stretch factors are shown in Fig. 11. We observe that the stretch factor of GDSTR-3D increases fast with the decrease of d, because the connections between nodes become more random, leading to more overlap between convex hulls. Consequently, a non-optimal branch is more frequently chosen to establish a routing path, resulting in a higher stretch factor. On the other hand, TR only increases stretch factor moderately due to the lower nodal density.

4.2 Overhead and Stability

TR does not require any global information to be stored by individual nodes. Each node only collects its 1-hop neighbor information, which is proportional to nodal degree, i.e., the density of the network, which is usually regarded as constant for a given network setting. The information necessary for greedy forwarding is simply the destination ID and location. The information needs for trace-routing is the plane defined by the destination, local minimum, and another arbitrary node. The computation overhead includes the identification of boundary neighbors and the next hop on the trace, which is proportional to the nodal degree as well. So

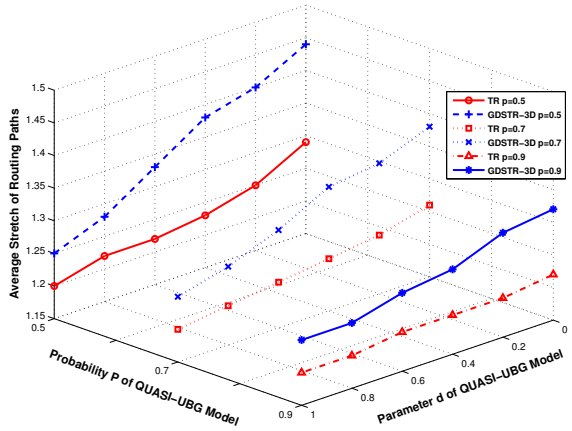


Figure 11: Stretch under Quasi-UDG.

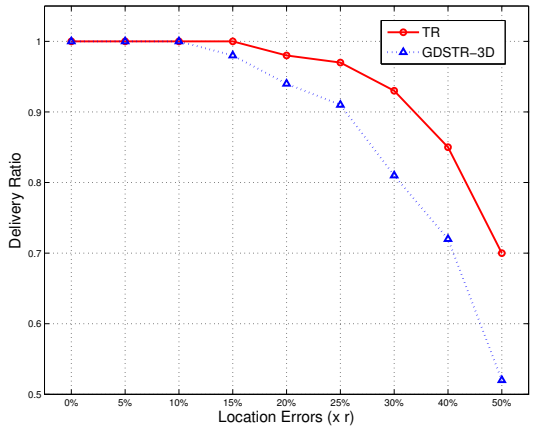


Figure 12: Delivery ratio against location errors.

both communication and computation overhead are bounded by a small constant (i.e., nodal degree). GDSTR-3D constructs two convex hull-based trees, where each node aggregates the locations of its children by using 2D convex hulls. Its time complexity and communication cost are dominated by the computation of 2D convex hulls (i.e., $O(n \log n)$), while the per node storage cost can be close to a constant after compression.

Location information is required in both TR and GDSTR-3D, which is, however, often subject to errors in practical sensor network settings. Fig. 12 depicts the results of TR and GDSTR-3D under different localization errors. We consider a path fails if its stretch factor exceeds 6.0. The delivery ratio of GDSTR-3D drops much faster than TR, because the location errors on individual nodes are aggregated in establishing the convex hull tree, increasing the chance to search into an incorrect tree branch. Since TR uses only local information only, the localization errors do not spread out. Therefore TR exhibits desired stability against localization errors. For example, under 30% errors, it still keeps its delivery ratio over 94%, while GDSTR-3D can deliver merely 81% of data.

4.3 Network Dynamics

Since TR does not rely on any global network structure, it is adaptive to network dynamics (e.g., sensors' movement). Here we evaluate the performance of TR, GDSTR-3D and HVE in mobile 3D sensor networks. We assume the holes are obstacles in the field where sensor nodes cannot access, and assume a simple random waypoint mobility model. More specifically, each node chooses a random position within a distance of ρ as its new position, where ρ is the maximum moving range during a time interval t. We let $\rho = 0.2r$ in our simulations. In each time interval, we measure the delivery ratios of three algorithms based on $10,000$ pair of randomly chosen nodes. Similar to the above discussion, a path is deemed unsuccessful if its stretch factor exceeds 6.0. The simulation lasts for 20 time intervals. As can be seen in Fig. 13, TR achieves a stable delivery ratio which is very close to 1 as expected, since the geometric shape of the network remains the same, even though the nodes change their positions. The failed routes, which are rare, are caused by the irregularity distribution of boundary nodes that results in very low node density in a local area. In a sharp contrast, the delivery ratios of GDSTR-3D and HVE decreases dramatically after 9 intervals, because the convex hull tree for GDSTR-3D and tetrahedron mesh for HVE become outdated, providing wrong routing information.

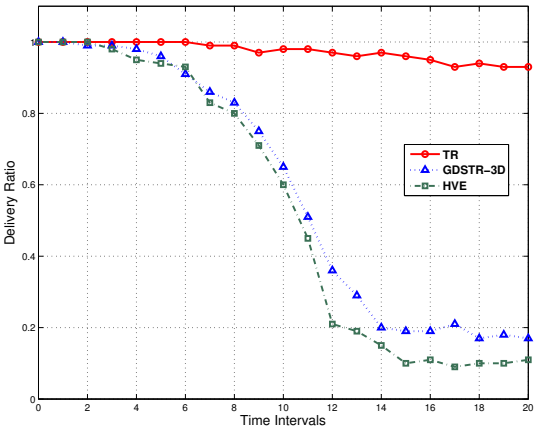


Figure 13: Delivery ratio under network dynamics.

5. EXPERIMENTS

We have also implemented the TR algorithm on Crossbow MICAZ motes (MPR2400CA). We set up a small testbed with 98 motes, where the maximum transmission range is set to around 4.5 inches for easy configuration of network topology. We place the motes uniformly around the boundary of a cubic hole of $20 \times 20 \times 20$ inches and manually measure the relative coordinates of each mote. The distance between neighboring motes is 4 inches and the nodal degree is around 4, as shown in Fig 14. Since all motes are boundary nodes, we skip boundary identification in our implementation. Each mote keeps a table to store its neighboring motes' coordinates, which is an very small array (4×4 in size), which contains node ID and 3D-coordinates. The exchange of neighboring information is done during the bootstrap phase, which takes less than 20 seconds. The TR algorithm involves mostly vectors operations, such as cutting plane identification, intersection between line segment and plane, and calculation of angle between two vectors, which are very convenient to implement on MICAZ motes. A data packet contains a dummy data segment (of 512 bytes), the IDs and coordinates of the source and destination, and a flag to indicate current routing method (i.e., greedy or trace-routing). If trace-routing is employed, the data packet also includes the ID and coordinates of the local minimum node, and the cutting plane (represented as $Ax + By + Cz + D = 0$ where A, B, C and D are computed only once at the local minimum node based on the coordinates of

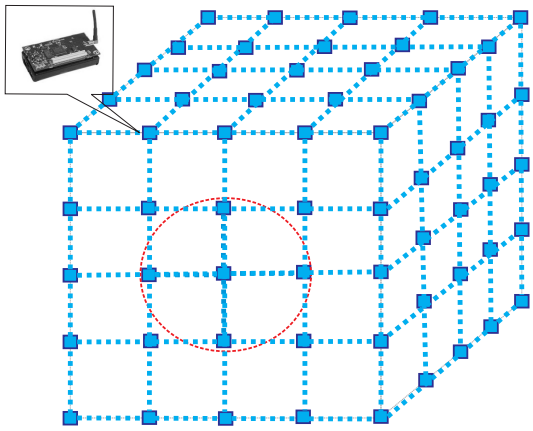


Figure 14: Experiment setup based on MicaZ motes.

the local minimum, the destination and an arbitrary neighboring node). The total overhead per packet is 52 bytes. We select 40 pairs of motes on different sides of the hole and transmit 50 packets for each route. All data packets are successfully delivered from source to destination in our experiment as expected, with an average stretch factor of 1.06.

6. CONCLUSION AND FUTURE WORKS

We have proposed *trace-routing*, a distributed and deterministic routing algorithm with constant storage, communication and computation overhead for 3D wireless sensor networks. Its basic idea is to construct a virtual cutting plane that intersects boundary surface to yield a trace, along which a routing path with guaranteed delivery can be established. We have proven the correctness of trace-routing under both continuous and discrete settings. We have implemented trace-routing on Crossbow sensors and carried out extensive simulations to evaluate its routing efficiency.

Acknowledgements.

Hongyi Wu is partially supported by NSF CNS-1018306 and CNS-1320931. Miao Jin is partially supported by NSF CCF-1054996, CNS-1018306 and CNS-1320931.

7. REFERENCES

[1] J. Allred, A. B. Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, "SensorFlock: An Airborne Wireless Sensor Network of Micro-Air Vehicles," in *Proc. of SenSys*, pp. 117–129, 2007.

[2] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications," *IEEE Network, Special Issue on Wireless Sensor Networking*, vol. 20, no. 3, pp. 12–18, 2006.

[3] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, "Low-Connectivity and Full-Coverage Three Dimensional Networks," in *Proc. of MobiHOC*, pp. 145–154, 2009.

[4] X. Bai, C. Zhang, D. Xuan, and W. Jia, "Full-Coverage and K-Connectivity (K=14, 6) Three Dimensional Networks," in *Proc. of INFOCOM*, pp. 388–396, 2009.

[5] C. Liu and J. Wu, "Efficient Geometric Routing in Three Dimensional Ad Hoc Networks," in *Proc. of INFOCOM*, pp. 2751–2755, 2009.

[6] T. F. G. Kao and J. Opatmy, "Position-Based Routing on 3D Geometric Graphs in Mobile Ad Hoc Networks," in *Proc. of The 17th Canadian Conference on Computational Geometry*, pp. 88–91, 2005.

[7] J. Opatrny, A. Abdallah, and T. Fevens, "Randomized 3D Position-based Routing Algorithms for Ad-hoc Networks," in *Proc. of Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, pp. 1–8, 2006.

[8] R. Flury and R. Wattenhofer, "Randomized 3D Geographic Routing," in *Proc. of INFOCOM*, pp. 834–842, 2008.

[9] F. Li, S. Chen, Y. Wang, and J. Chen, "Load Balancing Routing in Three Dimensional Wireless Networks," in *Proc. of ICC*, pp. 3073–3077, 2008.

[10] J. Zhou, Y. Chen, B. Leong, and P. Sundaramoorthy, "Practical 3D Geographic Routing for Wireless Sensor Networks," in *Proc. of SenSys*, pp. 337–350, 2010.

[11] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing Algorithms for Delay-insensitive and Delay-sensitive Applications in Underwater Sensor Networks," in *Proc. of MobiCom*, pp. 298–309, 2006.

[12] W. Cheng, A. Y. Teymorian, L. Ma, X. Cheng, X. Lu, and Z. Lu, "Underwater localization in sparse 3d acoustic sensor networks," in *Proc. of INFOCOM*, pp. 798–806, 2008.

[13] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," in *Proc. of Third Workshop Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 48–55, 1999.

[14] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of MobiCom*, pp. 1–12, 2001.

[15] E. Kranakis, H. Singh, and J. Urrutia, "Compass Routing on Geometric Networks," in *Proc. of CCCG*, pp. 51–54, 1999.

[16] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric Ad-hoc Routing: Theory and Practice," in *Proc. of The 22nd ACM Symposium on the Principles of Distributed Computing*, pp. 63–72, 2003.

[17] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case Optimal and Average-case Efficient Geometric Ad-hoc Routing," in *Proc. of MobiHOC*, pp. 267–278, 2003.

[18] B. Leong, S. Mitra, and B. Liskov, "Path Vector Face Routing: Geographic Routing with Local Face Information," in *Proc. of ICNP*, pp. 147–158, 2005.

[19] H. Frey and I. Stojmenovic, "On Delivery Guarantees of Face and Combined Greedy-face Routing in Ad Hoc and Sensor Networks," in *Proc. of MobiCom*, pp. 390–401, 2006.

[20] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks," in *Proc. of INFOCOM*, pp. 1719–1727, 2009.

[21] C. Papadimitriou and D. Ratajczak, "On A Conjecture Related to Geometric Routing," *Theoretical Computer Science*, vol. 344, no. 1, pp. 3–14, 2005.

[22] P. Angelini, F. Frati, and L. Grilli, "An Algorithm to Construct Greedy Drawings of Triangulations," in *Proc. of The 16th International Symposium on Graph Drawing*, pp. 26–37, 2008.

[23] T. Leighton and A. Moitra, "Some Results on Greedy Embeddings in Metric Spaces," in *Proc. of The 49th IEEE*

Annual Symposium on Foundations of Computer Science, pp. 337–346, 2008.

[24] R. Kleinberg, "Geographic Routing Using Hyperbolic Space," in *Proc. of INFOCOM*, pp. 1902–1909, 2007.

[25] A. Cvetkovski and M. Crovella, "Hyperbolic Embedding and Routing for Dynamic Graphs," in *Proc. of INFOCOM*, pp. 1647–1655, 2009.

[26] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. Gu, "Greedy Routing with Guaranteed Delivery Using Ricci Flows," in *Proc. of IPSN*, pp. 121–132, 2009.

[27] R. Flury, S. Pemmaraju, and R. Wattenhofer, "Greedy Routing with Bounded Stretch," in *Proc. of INFOCOM*, pp. 1737–1745, 2009.

[28] S. Durocher, D. Kirkpatrick, and L. Narayanan, "On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks," in *Proc. of International Conference on Distributed Computing and Networking*, pp. 546–557, 2008.

[29] Y. Wang, C.-W. Yi, and F. Li, "Delivery Guarantee of Greedy Routing in Three Dimensional Wireless Networks," in *Proc of International Conference on Wireless Algorithms, Systems, and Applications*, pp. 4–16, 2008.

[30] S. S. Lam and C. Qian, "Geographic Routing with Low Stretch in d-dimensional Spaces," in *Proc. of SIGMETRICS*, pp. 257–268, 2011.

[31] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, "Deterministic Greedy Routing with Guaranteed Delivery in 3D Wireless Sensor Networks," in *Proc. of MobiHoc*, pp. 1–10, 2011.

[32] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks," in *Proc. of ICDCS*, pp. 744–753, 2010.