Geometric Spanners for Routing in Mobile Networks

Jie Gao, *Member, IEEE*, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu

***Abstract*—We propose a new routing graph, the restricted Delaunay graph (RDG), for mobile ad hoc networks. Combined with a node clustering algorithm, the RDG can be used as an underlying graph for geographic routing protocols. This graph has the following attractive properties: 1) it is planar; 2) between any two graph nodes there exists a path whose length, whether measured in terms of topological or Euclidean distance, is only a constant times the minimum length possible; and 3) the graph can be maintained efficiently in a distributed manner when the nodes move around. Furthermore, each node only needs constant time to make routing decisions. We show by simulation that the RDG outperforms previously proposed routing graphs in the context of the Greedy perimeter stateless routing (GPSR) protocol. Finally, we investigate theoretical bounds on the quality of paths discovered using GPSR.**

***Index Terms*—Geographical routing, spanners, wireless ad hoc networks.**

I. Introduction

An AD HOC network consists of a collection of mobile communication nodes. Any two nodes within a certain distance of each other can communicate directly.[1] There is no centralized control or other fixed infrastructure. Each mobile node can operate as a router, relaying packets for other nodes. The nodes may move continuously and turn themselves on/off arbitrarily. The constantly changing network topology makes routing in *ad hoc* networks difficult.

Many routing protocols have been proposed for mobile networks [1]. Among them geographical forwarding, being a simple and scalable routing scheme, has attracted a lot of interests in recent years. The nodes can obtain their location information by either global positioning system (GPS) or localization algorithms [2], [3]. Simple geographical forwarding combined with GLS (scalable location service [4]) compares favorably with dynamic source routing (DSR) [5], [6]. It delivers more packets and consumes fewer network resources. Furthermore, the performance of GLS degrades gracefully as nodes fail and restart, and is relatively insensitive to node speeds [4]. The reason for the scalability and efficiency of location-based routing algorithms is that they adopt *local algorithms*, i.e., each node makes the decision on which node to forward the packet to, based solely on the location of itself, the neighboring nodes, and the destination. Local algorithms are lightweight, robust, and distributed in nature. In contrast, the shortest-path-based algorithms require the knowledge of the complete network topology, whose maintenance cost is quadratic in the size of the network—each change in edge or node status (nodes switch on/off/sleep) may trigger routing table update in a large portion of the network. Location-based routing algorithms reduce such overhead.

In a geographic forwarding scheme, a source node first acquires the location of the destination node it wants to communicate with, then forwards the packet to a neighbor closer to the destination. This process is repeated until the packet reaches the destination. Thus, a path is found via a series of local decisions rather than flooding. However, geographic forwarding methods suffer from the so called *local minimum phenomenon*, in which a packet gets stuck at a node that does not have a closer neighbor to the destination, even though the source and destination are connected in the network. One technique to deal with this problem, proposed by Bose *et al.* [7] and independently Karp and Kung [6], is to maintain a planar subgraph of the underlying connectivity. When a packet is stuck at a node, the protocol will route the packet around a face of the graph to get out of the local minimum. Karp and Kung also proposed a routing protocol, the greedy perimeter stateless routing (GPSR) protocol that guarantees the delivery of the packet if a path exists. The advantage of GPSR over other routing protocols is that forwarding decisions are made using local information only; there is no need to maintain routing tables or make global broadcasts.

Two planar subgraphs, the *relative neighborhood graph* (RNG) and the *Gabriel graph* (GG), were proposed to solve the local minima problem in geographical routing. Both of them are based on local geometric conditions and can be computed efficiently. While the algorithms perform well when each individual node's visible range is large and nodes are uniformly or randomly distributed, they do not perform as well for more general node distributions. In particular, the GG and RNG are not good spanners: nodes that can be reached via a path with few hops might become far apart in the GG or RNG [8]. This fact limits the quality of paths even if we use globally optimum routing methods on these subgraphs. In this paper, we use the *stretch factor* to capture this aspect of path quality. Roughly speaking, the stretch factor of a subgraph G' of a graph G measures the worst case ratio between the length of a shortest

Manuscript received October 15, 2003; revised August 17, 2004.

J. Gao was with the Department of Computer Science, Stanford University, Stanford, CA 94305 USA. She is now with the Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jgao@ist.caltech.edu).

L. J. Guibas and A. Zhu are with the Department of Computer Science, Stanford University, Stanford, CA 94305 USA (e-mail: guibas@cs.stanford.edu; anzhu@cs.stanford.edu).

J. Hershberger is with Mentor Graphics Corporation, Wilsonville, OR 97070 USA (e-mail: john_hershberger@mentor.com).

L. Zhang was with Compaq Systems Research Center, Palo Alto, CA. He is now with the Information Dynamics Laboratory, Hewlett Packard Laboratories, Palo Alto, CA 94304 USA (e-mail: l.zhang@hp.com).

Digital Object Identifier 10.1109/JSAC.2004.837364

[1]We use a simplified model that all nodes have the same transmission range and two nodes can directly communicate if their distance is within the transmission range. The interference of the wireless channels and the presence of obstacles, though being important issues, are not considered in this paper.

0733-8716/$20.00 © 2005 IEEE

path in G' to the length of the shortest path with the same endpoints in G. A subgraph with a constant stretch factor is called a *spanner*.

We present a new routing graph, the *restricted Delaunay graph* (RDG), that has nice theoretical guarantees on the stretch factor of routing paths. In particular, the RDG has paths with Euclidean and topological length only a constant factor longer than the length of the optimal path. Our routing graph can be efficiently computed and maintained in a completely distributed and local manner. The total communication cost (message exchange) for its construction is only linear in the total number of nodes. Under topological changes (edge insertion or deletion), only nodes within two hops need to be updated. The update cost is $O(1)$ per node per topological change. In addition to presenting a rigorous theoretical analysis, we also demonstrate by simulation that GPSR on the RDG finds routes of substantially better quality as compared with the GG or RNG, under both uniform and multimodal distributions of the points.

To define our graph, we first group nodes into clusters. Each cluster has a *clusterhead*, and nearby clusters are connected via *gateway nodes*. For a node u to send a packet to a nonneighbor node v, u first forward the packet to its clusterhead; the packet is then forwarded on the RDG defined only on clusterheads and gateways until it reaches some clusterhead or gateway that is visible to v. We use a clustering algorithm to guarantee that each clusterhead/gateway has only a constant number of neighbors on RDG [9]. This simplifies forwarding during routing. For instance, in [6], the greedy geographic forwarding is done by examining all neighboring nodes in order to skip short edges in the graph. This process is expensive when nodes are densely distributed. In our routing graph, we perform greedy geographic forwarding by considering only the adjacent nodes in the RDG, therefore, reduce the complexity significantly. The clustering algorithm also improves the behavior of GPSR. GPSR based on the GG or RNG may traverse a short boundary that consists of a dense sequence of nodes; but boundaries in the RDG have only constant density. We also investigate the tradeoff between scaling and the spanning property, and the efficiency of clusterhead changes.

The rest of the paper is organized as follows. Section II covers related work. Section III gives a detailed description of the RDG and proves the spanning property. Section IV deals with the distributed implementation of the RDG. Section V proves theoretical bounds on the length of the actual routing paths under certain circumstances. We compare the simulation results for GPSR on the RDG versus GPSR on the RNG in Section VI, and finally, Section VII concludes by discussing various other aspects of the RDG.

II. Related Work

Clustering is used in many routing protocols in mobile networks [10]–[13]. One class of clustering algorithms is based on the minimum connected dominating set (MCDS) [14], [15]. Das *et al.* [14] proposed a MCDS routing protocol that uses a $\log n$-approximation of the minimum connected dominating set. Wu *et al.* used a distributed algorithm to compute the connected dominating set; however, this could perform badly in the worst case ($O(n)$-approximation) [15]. Another class of clustering algorithms started from the lowest-ID cluster algorithm proposed by Ephremides *et al.* [12], [13]. A similar idea also led to the max-min D-clustering scheme proposed by Amis *et al.* [10]. Chiang *et al.* [11] proposed a least cluster change clustering (LCC) algorithm that tries to minimize clusterhead changes under motion. However, there is no guarantee on the quality of the clustering. They also proposed a clusterhead gateway switch routing (CGSR) protocol that uses routing tables; yet maintaining the clustering with a routing table is expensive in a mobile setting. For mobile nodes, Gao *et al.* [9] and Hershberger [16] proposed constant approximation clustering algorithms with efficient maintenance schemes. The algorithm by Hershberger [16] requires global information. In this paper, we make use of the algorithm in [9] because of the guaranteed good quality of the clusters and the efficient distributed maintenance under motion.

Spanner graphs have been heavily studied in computational geometry [8]. The Delaunay triangulation is known to be a planar spanner [17]–[19]. However, little is known about restricted spanner graphs, where only edges shorter than one are allowed. The preliminary version of this paper [20] is the first paper to propose a planar spanner in both Euclidean and topological distance measures that can be constructed and maintained in a distributed and local manner as the nodes move around. Li *et al.* [21] proposed a k-localized Delaunay graph for static networks and proved it is planar if $k \geq 2$ and has a constant stretch factor for Euclidean length only. In a later paper, Alzoubi *et al.* [22] used ideas similar to those of [20] and combined the k-localized Delaunay graph with the dominating set approach to prove a constant stretch factor for topological distance on their local Delaunay (LDel) graph. Again, they considered static networks only. None of these authors studied in detail how to maintain spanner graphs in a distributed fashion when the nodes move around.

III. Routing Graph With Constant Stretch Factor

We assume that two mobile nodes can communicate with each other directly if their separation is no larger than one. We call two such nodes *visible* to each other. The unit-disk graph $\mathcal{G} = (V, E)$ is defined as follows: the vertex set V is the set of all the mobile nodes, and an edge uv is in E if and only if u and v are visible to each other. The neighborhood of u, denoted by $N(u)$, is the set of the nodes visible to u (including u itself). If we assume $|V| = n$, then there may be $\Theta(n^2)$ edges in $\mathcal{G}$. For any two nodes u and v in V, denote by $\tau(u, v)(d(u, v))$ the length in hops (in Euclidean distance) of the topological (Euclidean) shortest path connecting u and v in $\mathcal{G}$. For a subgraph G of $\mathcal{G}$, define $\tau_G(u, v)$ and $d_G(u, v)$ to be the same quantities in G. Then, G has topological (Euclidean) stretch factor at most C if for any pair of nodes $u, v, \tau_G(u, v) \leq C \cdot \tau(u, v)(d_G(u, v) \leq C \cdot d(u, v))$. The stretch factor measures the quality of the subgraph. One of the major goals of this paper is to construct a sparse planar subgraph G with constant stretch factor. This graph G can serve as a routing graph in ad hoc networks.

Our construction consists of two phases. First, we make use of the hierarchical clustering algorithm in [9] to select a small subset of V, called the *clusterheads*, so that each node in V can communicate directly to a clusterhead. Each nonclusterhead node in V (called a *client*) is assigned to a unique clusterhead visible to it. We also identify those pairs of clusterheads that may communicate to each other via their clients. For each such pair, we pick one pair of clients, called *gateways*, that enable such communication. This reduces routing in $\mathcal{G}$ to routing between clusterheads and gateways. Second, we form a planar routing graph on clusterheads and gateways by applying a local rule, called the *restricted Delaunay edge* rule. The graph produced this way is called RDG. Routing between clusterheads and gateways is then done on the RDG. Therefore, our final routing graph R is the union of RDG and the edges that connect clients to clusterheads.

Our routing graph has the following properties.

- The RDG is a planar graph. No two edges cross each other in the graph.
- Graph R has constant stretch factor, under both topological and Euclidean measures, compared with the original communication graph G. If there exists a path in $\mathcal{G}$ with length ℓ between two nodes, then there is a path in R with length $C \times \ell$ for some constant $C > 0$, where the length can be either topological or Euclidean distance.
- Graph R can be efficiently computed and maintained in a completely distributed and local manner. The total communication, (i.e., the number of messages exchanged) and computation cost for the construction is $O(n)$. Under topological changes (edge insertion or deletion), only nodes within two hops need to be updated. The update cost and the message exchange is $O(1)$ per node per topological change.

In the rest of this section, we will briefly describe the clustering algorithm in [9] and present the RDG.

A. Mobile Clustering

The goal of clustering is to select a subset of nodes as *clusterheads* such that the rest of the nodes are visible to at least one of the clusterheads. While any clustering algorithm can be used in the first stage, the algorithm developed in [9] is used here because we need some special properties of the clustering algorithm to achieve good properties on the routing graph.

The one-level clustering algorithm works as follows. Assume a random ordering on the unique IDs of the nodes, and let each node nominate the node with the highest ordered ID in its visible range.[2] All nominated points are clusterheads. A cluster is formed by a clusterhead and all the nodes that nominated it. This one-level algorithm was first proposed by Ephremides *et al.* [12], but without theoretical analysis. In [9], the method is rigorously analyzed and extended to a hierarchical algorithm that achieves a constant approximation factor in expectation. □

The hierarchical algorithm makes use of the one-level algorithm and proceeds in a number of rounds. The basic idea is that instead of considering all nodes in its visible range, each node gradually grows its visible range and selects clusterheads among nodes in the restricted visible range. Only clusterheads selected in one round will participate in the clusterhead selection process in the next round. More precisely, at round 0, all nodes are clusterheads and participants. At each round every participant (clusterhead produced by the previous round) selects a new clusterhead out of the participants within a larger visible range by using the basic one-level algorithm. The size of the visible range used in round i is $2^i/\lg n$. The hierarchical algorithm terminates after $\log\log n - 1$ rounds. A cluster is defined by a final clusterhead and all the nodes that directly or indirectly nominated it. Gao *et al.* showed that all nodes in a cluster are visible to the clusterhead, and the number of final clusterheads is only a constant factor more than the minimum possible [9]. Therefore, the clusterheads have constant density—any unit disk only covers a constant number of clusterheads. This is due to the fact that the optimal solution has density at most 6 (a simple greedy algorithm[3] produces a feasible clustering with density at most 6, the optimal solution can only do better).

Theorem 3.1: The number of clusterheads in any unit disk is $O(1)$ in expectation.

To enable different clusters to communicate with each other, we introduce *gateways* [12]. These are nodes that link two clusters. For each clusterhead p, define the cluster $C(p)$ centered at node p to be the set of points that nominated p and p itself. Note that one node can participate in two clusters, if it nominates another node as its clusterhead, and at the same time compared with nominated by others to be a clusterhead. Node x's clusterhead is denoted by c_x. For a pair of clusterheads (c_1, c_2), if there exists a pair of nodes $p_1 \in C(c_1), p_2 \in C(c_2)$ such that p_1 and p_2 are visible to each other, we define p_1 and p_2 to be *gateway* nodes. Note that p_1 and p_2 might be clusterheads already, in which case they remain clusterheads. Between each pair of overlapping or adjacent clusters, only one pair of gateway nodes is maintained at any time. We describe the maintenance of clusterheads and gateways for mobile nodes in Section IV. From Theorem 3.1, we can also derive the following fact.

Corollary 3.2: The number of clusterheads and gateways in any unit disk in the plane is $O(1)$ in expectation.

Proof: If clusterhead c_2 has a pair of gateway nodes with clusterhead c_1, c_2 must be at most distance 3 away from c_1. So the number of clusterheads that can form gateways with c_1 is at most a constant. That is, there are at most a constant number of gateways in any unit disk. The number of clusterheads and gateways in any unit disk is also bounded by $O(1)$ in expectation. □

The hierarchical algorithm provides a theoretical bound that holds for *any* distribution of nodes in the plane. In reality, distributions that cause bad clustering quality appear very rarely and the one-level algorithm actually works pretty well for practical situations. In our simulations, we only use the one-level clustering algorithm described above.

B. Restricted Delaunay Graph (RDG)

Our routing graph R includes the edges that connect each client to its clusterhead. In addition, we build a planar graph

[2]There is a way to permute the order of the IDs such that every node gets a fair chance of being a clusterhead.

[3]The greedy algorithm works as follows. Select an arbitrary node as a clusterhead. Delete all the nodes covered by the selected clusterhead. Repeat this process until all the nodes are covered.

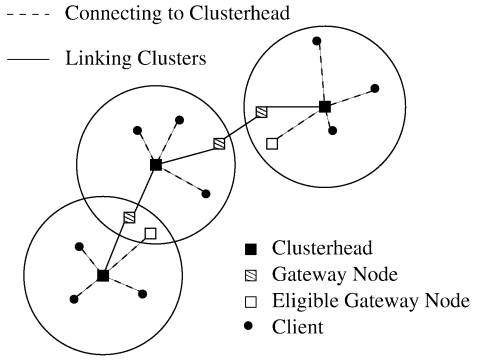


Fig. 1. Example of linked cluster organization of a mobile network.

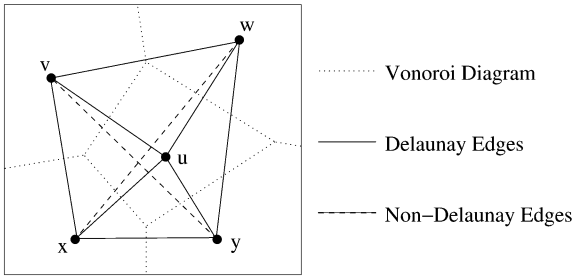


Fig. 2. Voronoi diagram and Delaunay triangulation of a set of points.

among clusterheads and gateways. Fig. 1 shows an example of such a routing graph.

The routing of a packet from u to v (if v is not directly reachable) is realized by first sending the packet to u's clusterhead, then forwarding the packet among clusterheads and gateways until it reaches a node visible to v, which forwards the packet to v. We design an RDG for connecting clusterheads and gateways, similar to the GG and RNG used for all the nodes in GPSR [6]. The main difference is that the RDG provides theoretical guarantees on the Euclidean and topological stretch factors, while the GG and RNG do not (Section V). In the rest of this section, we concentrate on the unit-disk graph whose nodes are the clusterheads and gateways G_{CG}. Since the definition of the RDG is independent of the clustering algorithm, we will describe the graph on a set of points, but the reader should keep in mind that the graph is computed on clusterheads and gateways, rather than on the full node set V.

1) Voronoi Diagram, Delaunay Triangulation, and RDG: For a set of point sites in the plane, the Voronoi diagram partitions the plane into convex polygonal faces such that all points inside a face are closest to only one site. The Delaunay triangulation is the dual graph of the Voronoi diagram, obtained by connecting the sites whose faces are adjacent in the Voronoi diagram. For an edge xy, there is an *empty-circle* rule to determine whether xy is a Delaunay edge: xy is a Delaunay edge if and only if there exists a circle that contains no other points except x, y. Fig. 2 shows an example of a Voronoi diagram and Delaunay triangulation of a set of points.

These classical geometric structures have numerous applications [23]. The Delaunay triangulation is known to be a good spanner of the complete graph, measured by the Euclidean distance [18], [19]. However, we cannot use this graph directly in our setting because: 1) the Delaunay triangulation may have long edges, while we are only allowed to connect points within distance 1 and 2) the empty-circle rule is a global rule and is not suitable for local computation. To deal with those two problems, we define the RDG and show that it has good spanning properties and is easy to maintain locally.

Definition 3.3: A restricted Delaunay graph of a set of points in the plane is a planar graph and contains all the Delaunay edges with length ≤ 1 (called short Delaunay edges).

Notice that the RDG always exists and is not necessarily unique—it may contain additional edges beyond the short Delaunay edges. By its planarity, we also know that a RDG is sparse, i.e., has linearly many edges in terms of the number of nodes. In the following, we will show that any RDG has nice spanning properties.

2) Spanning Properties of R: The Euclidean spanning property of the Delaunay triangulation was first proved in [18] and later improved in [19]. We extend the proof in [18] to show the Euclidean distance spanning property of the graph S, which is a subgraph of a unit-disk graph $\bar{G} = (\bar{V}, \bar{E})$ and contains only the short Delaunay edges. Then, an RDG graph is also an Euclidean distance spanner, since it necessarily contains graph S.

Lemma 3.4: For any $u, v \in \bar{V}, d_S(u, v) < C_1 \cdot d(u, v)$, where $C_1 = (1 + \sqrt{5}/2)\pi \approx 5.08$, i.e., S is an Euclidean spanner graph with stretch factor of at most 5.08.

Proof: It suffices to prove that for any two nodes $u, v \in \bar{V}$, if their Euclidean distance is $\ell \leq 1$, then there exists a path in RDG connecting them whose total Euclidean length is at most $C_1 \cdot \ell$. We can use the following spanning property proved for regular Delaunay triangulations by Dobkin *et al.* [18]: for any two nodes u, v, there exists a path in the Delaunay triangulation that lies entirely inside the circle with uv as the diameter, and the path length is no more than $(1 + \sqrt{5}/2)\pi \cdot \ell$. For any two points in the circle with uv as the diameter, their distance is at most $\ell \leq 1$. Therefore, all the edges in the path constructed in [18] are short Delaunay edges, which all exist in S. □

While the above lemma shows that RDGs are good Euclidean spanners, an RDG is not necessarily a good topological spanner. A counterexample can be constructed, where there is a direct path between two nodes yet in the RDG the number of hops of the shortest path is $\Theta(n)$. However, if nodes are distributed with constant density, i.e., there are $O(1)$ nodes in any unit circle in the plane, then we can also show that the topological stretch factor is bounded. Fortunately, the graph G_{CG} has constant density by Corollary 3.2.

Lemma 3.5: An RDG is a topological spanner graph with constant stretch factor. That is, for any two nodes u, v in $G_{\mathrm{CG}}, \tau_{\mathrm{RDG}}(u, v) \leq C_2 \cdot \tau(u, v)$ for some constant $C_2 > 0$.

Proof: Since G_{CG} has constant density, in the Proof of Lemma 3.4, there are at most $O(1)$ points in the circle with uv as the diameter. Thus, the path in the RDG has a constant number of intermediate nodes. That is, the RDG is a topological spanner graph with constant stretch factor. □

In addition, our routing graph R is an Euclidean and topological spanner.

Theorem 3.6: Graph R is an Euclidean and topological spanner with respect to the underlying unit-disk graph.

Proof: Suppose the (either Euclidean or topological) shortest path between u and v is $P : u_1 = u, u_2, \ldots, u_{k+1} = v$

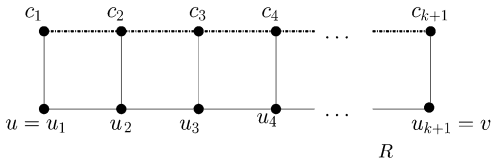


Fig. 3. Spanner property of routing graph R.

```
E(u) := {uv | uv ∈ T(u)}
For each edge uv in E(u)
    For each w in N(u)
        If (u, v ∈ N(w) and uv ∉ T(w)) then
            delete uv from E(u)
```

Fig. 4. Pseudocode for resolving inconsistency.

and the clusterhead of u_i is c_i. Since node u_i and u_{i+1} are visible to each other, there must exist a pair of gateway nodes between clusterheads c_i and c_{i+1}, i.e., $\tau_{G_{\mathrm{CG}}}(c_i, c_{i+1}) \leq 3$ (Fig. 3). From Lemma 3.5, there exists a path P_i in the RDG whose length is at most $C_2 \cdot \tau_{G_{\mathrm{CG}}}(c_i, c_{i+1})$. We define the path P' to be the union of P_i and the edges $u_1c_1, u_{k+1}c_{k+1}$. Then, $\tau_R(u, v) \leq 2 + \sum_{i=1}^{k} C_2 \cdot \tau_{G_{\mathrm{CG}}}(c_i, c_{i+1}) \leq 3C_2 \cdot k + 2$.

The Euclidean spanner property follows from the constant density of the clusterheads and gateways. Basically, all the c_i's are in a region whose area is linear to the Euclidean length of P. Due to the constant density argument, the number of clusterheads and gateways inside the region is also linear to the length of P. So the length of the path P' is only a constant times the length of P. □

We have shown that our routing graph has constant stretch factor for both Euclidean and topological distances. In practice, it is expensive to find the shortest path in a routing graph. Geographic forwarding is preferable because of its simplicity. We will prove some theoretical bounds on the length of the actual paths used by geographic forwarding in Section V. In addition, our experimental results show that our routing graph performs better than the RNG or GG under geographic forwarding (Section VI).

IV. Maintaining the Routing Graph

In this section, we discuss the maintenance of the routing graph in the distributed setting. The challenge here is that each node only has a fixed communication range and only performs local computation. We aim to design an algorithm that enables each node to maintain the relevant part of the routing graph efficiently and consistently, with only the knowledge of its neighbors. For the maintenance of clusterheads, we use the algorithm described in [9] and refer to that paper for details. Here, we will describe the maintenance of RDGs and gateway nodes.

A. Maintaining an RDG

According to Lemma 3.5, any RDG has the desired spanning property. We will describe a distributed algorithm that maintains an RDG as nodes move. Each node u maintains a set of edges $E(u)$ incident to u and those edges satisfy that: 1) each edge in $E(u)$ is short, i.e., of length < 1; 2) the edges are consistent, i.e., the edge uv is in $E(u)$ if and only if it is in $E(v)$; 3) the graph obtained is planar, i.e., no two edges cross; and (4) all the short Delaunay edges are guaranteed to be in the union of $E(u)$'s.

The algorithm works as follows. First, each node u acquires the position of its one-hop neighbors $N(u)$ among the clusterheads and gateways, and compute their Delaunay triangulation (including u itself), denoted by $T(u)$. Since $T(u)$ is computed only on $N(u)$, the edges we obtain are a superset of the short Delaunay edges and some of them might be globally non-Delaunay edges. Furthermore, the local Delaunay triangulations at different nodes might be inconsistent, i.e., an edge uv is in u's local graph but not in v's. Because of this inconsistency, the union of local graphs might not be planar although they are planar individually. To resolve these problems, in the second step, we perform a one-hop information exchange, i.e., each node sends its local Delaunay triangulation $T(u)$ to the clusterheads and gateways within one hop. Then, each node resolves the inconsistency by deleting edges that are not valid in its neighbors' local Delaunay triangulations. More precisely, each node executes the pseudocode shown in Fig. 4.

Now, we will argue that after the execution of the above pseudo code, all the $E(u)$'s satisfy the stated properties. The invariant the above pseudocode achieves is that for each visible pair u and v, the edge uv belongs to $E(u)$ if and only if $uv \in T(w)$ for all $w \in N(u) \cap N(v)$ (notice that $u, v \in N(u) \cap N(v)$ since u, v are mutually visible). If an edge uv is a short Delaunay edge, it has to be present in all the local graphs $T(w)$ for $w \in N(u) \cap N(v)$. Therefore, the properties 1), 2), and 4) hold. The following simple geometric fact shows that this one-hop information exchange suffices and there are no crossing edges.

Lemma 4.1: For two visible pairs uv and wx, if the edges uv and wx cross, then one of the four nodes sees all of the other three.

Proof: Assume that uv (of length ≤ 1) and wx (of length ≤ 1) intersect at a point p. By the triangle inequality, $|wp| + |up| \geq |uw|$ and $|vp| + |xp| \geq |vx|$. Summing these two equations, we have that $|uv| + |wx| \geq |uw| + |vx|$. Therefore, either uw or vx has length ≤ 1. Similarly, either ux or vw has length ≤ 1. No matter in which case, the endpoint shared by the two short edges sees all three other points. □

By Lemma 4.1, we now argue that no crossing exists in the final graph. Suppose that the edge $uv \in E(u)$ intersects the edge $wx \in E(w)$. Then, by the lemma, one of u, v, w, x, say u, sees all four nodes. Therefore, w must have received $T(u)$ when computing $E(w)$. According to Fig. 4, both uv and wx must be present in $T(u)$, contradicting the fact that $T(u)$ is a planar graph.

The procedure above could be expensive if $N(u)$ contains many nodes. Fortunately, it is not the case in our setting because we apply this algorithm on clusterheads and gateways. According to Corollary 3.2, those nodes have constant density. Therefore, $E(u)$ can be computed in $O(1)$ time for each u. Note that the computation is completely local and global flooding is not needed.

B. Maintaining Gateway Nodes

The maintenance of gateway nodes is similar to the method described in [12]. Essentially, every node sends its entire neighbor set to every neighboring node. However, each node

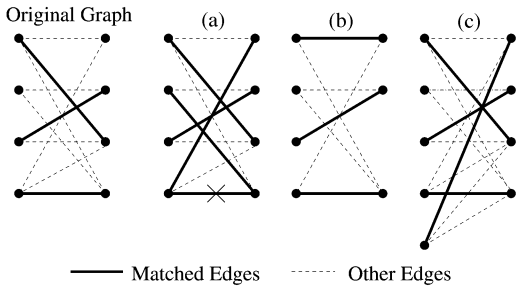


Fig. 5. Maximal matching in bipartite graph $B(c_1, c_2)$. (Left) Original graph. (a) Pair of nodes become invisible. (b) A node leaves the cluster. (c) A new node joins the cluster.

might take $O(D^2)$ time processing and making decisions about whether or not it should serve as a gateway, where D is the total number of nodes within two hops. While such a method is needed at the initial phase, it is not efficient as points are moving. We present here an algorithm to let clusterheads select gateways instead of each node making that decision. Note that changes to clusterheads and gateways occur only if the underlying unit-disk graph changes, i.e., when two nodes u and v become visible or invisible to each other. We show that when such an event happens, only the one-hop neighbors of u and v need proper update and the update time is constant per node.

For two clusterheads c_1 and c_2, we define a bipartite graph $B(c_1, c_2)$ with vertices $C(c_1) \cup C(c_2)$. The edge pq is in $B(c_1, c_2)$ if $p \in C(c_1), q \in C(c_2)$, and p is visible to q. The edges in the bipartite graph $B(c_1, c_2)$ represent all eligible gateway pairs between c_1 and c_2. To avoid storing all the edges in this graph, we only maintain a maximal matching $M(c_1, c_2)$ at c_1.[4] Fig. 5 shows such matchings. If pq is an edge in the matching, we call p is matched to q, or to $c_2(q \in C(c_2))$, or simply, p is matched. By maintaining maximal matchings, we can reduce storage needed to $O(D)$ and update time to $O(1)$ per node.

The property of maximal matching guarantees that if there is at least one edge in the bipartite graph, i.e., clusterheads c_1 and c_2 can be connected via gateways, all maximal matchings have to contain at least one edge, too. To maintain the maximal matching record, a clusterhead c_1 maintains the pair (p, c_2), where p is visible to c_1, p is matched to q, and $q \in C(c_2)$. For each matched node p (which may or may not be chosen as a gateway node), p maintains the pair (q, c_2), where p is matched to q, and $q \in C(c_2)$. At the beginning, after proper rounds of information propagation, each clusterhead pair would properly select a maximal matching from the bipartite graph (to make the matching consistent on both sides, we let the clusterhead with higher ID select the matching and inform the other clusterhead). We let the clusterhead with higher ID select a gateway pair out of the available matching. As points move around, if the previous selected gateways are no longer valid as indicated by the matching, the clusterhead would select another gateway pair out of the current matching.

When nodes move around, we want to maintain a maximal matching. We first describe how the neighborhood information

[4]A matching of a bipartite graph is a subgraph where each node has at most one edge. A maximal matching is a matching such that no edges can be added to the matching. A maximal matching can be constructed in a greedy way.

c_1	c_2	Nodes in $C(c_1)$ and matched to c_2
		Nodes in $C(c_1)$ and *not* matched to c_2
	c_3	Nodes in $C(c_1)$ and matched to c_3
		Nodes in $C(c_1)$ and *not* matched to c_3
	$\vdots$	$\vdots$
c_2	c_1	Nodes in $C(c_2)$ and matched to c_1
		Nodes in $C(c_2)$ and *not* matched to c_1
	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

Fig. 6. Organizing neighbors.

is organized inside a node u. To be more specific, each node v would propagate information by broadcasting an update entry of the following form

ID	c	c′	c_1	c_2	...

where the ID uniquely identifies v, c and c' are the IDs of the clusterheads that the clusters v belongs to (recall that a node may belong to two clusters), $c_1, c_2, \ldots$ are the IDs that the clusterheads v is matched to. Note that since clusterheads have constant density, each such entry is of constant size. Then, u would organize its neighbors' entries into a table that is indexed by a pair of clusterhead IDs (Fig. 6). The first index is the ID of a clusterhead that a neighbor belongs to, and the second index is the ID of a clusterhead that a neighbor is matched (not matched) to. This table enables $O(1)$ lookup time to find a neighbor whose clusterhead is c_i and currently matched (not matched) to c_j. For u, the indices of the table include only clusterheads within distance 3. Also, a neighbor v might appear in the table several times, but by the constant density argument, at each node the total number of pairs of indices in the table is a constant, each neighbor only appears a constant number of times. So the total storage at each node is still linear in the neighborhood size. The table and the maximal matching record are updated upon receiving any update entry from neighbors.

Changes of the maximal matching can only happen when two nodes begin or stop seeing each other, this may also cause one client in $C(c_i)$ to change clusterhead. We will discuss these situations separately.

1) When two nodes p and q begin to see each other $p \in C(c_1), q \in C(c_2)$, the change takes place if both p, q are not matched with respect to clusterheads c_1 and c_2. They become matched in $B(c_1, c_2)$ by adding c_2 and c_1 to the update entry respectively. Once the update entry is modified, a node would broadcast the entry to its neighbors (in the succeeding discussions we omit this step). If two nodes p, q stop seeing each other and they are matched before in $B(c_1, c_2)$, we need to find out if they can be matched with other nodes in the same bipartite graph. To do this, p and q would look into their neighbor set and find unmatched nodes in $C(c_2)$ and $C(c_1)$, respectively [Fig. 5(a)]. For example, p looks for a neighbor q' with c_2 as one of the clusterheads and q' is not matched to c_1.
2) When one node changes its clusterhead: This involves p disappearing in the original cluster and appearing in the new cluster. When p disappears in $C(c_1)$, if p is not matched at all, nothing needs to be done. If not then p

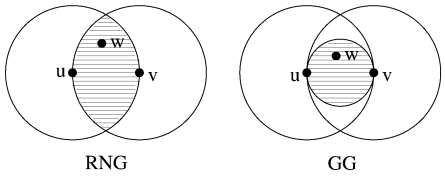


Fig. 7. RNG and GG.

needs to broadcast the clusterhead change to its neighbors. Notice that because of the constant density of clusterheads, p participates in at most a constant number of matchings in total. So a clusterhead change would only affect a constant number of nodes in the graph. Suppose p was matched to some node q in some $C(c_2)$, once q receives message from p about clusterhead changes, q needs to search its neighbors for potential matchings [Fig. 5(b)]. When p appears in $C(c_1)$, p needs to find among its neighbors nodes that belong to some other cluster $C(c_2)$ and are currently not matched to c_1. This can be done in a similar way as described in the previous situation [Fig. 5(c)].

To summarize, the RDG, as well as the routing graph R are light-weight structures that can be efficiently computed and maintained in a completely distributed and localized manner. The total communication and computation cost for the construction is only $O(n)$. Under topological changes (edge insertion or deletion), only nodes within two hops need to be updated. The update cost is $O(1)$ per node per topological change.

V. Quality Analysis of Routing Graphs

As shown in Section III-B2, our routing graph has bounded topological and Euclidean stretch factors. In the literature, there have been other ways proposed to construct good spanners. The most popular one is to partition the space around each point into cones with some fixed angles, and then connect the point to the nearest point in each cone. While such cone-based construction gives us good geometric spanners, they are generally nonplanar.

The RNG and the GG are two planar graphs used in [6]. The RNG is defined such that an edge (u, v) exists if there is no other node w whose distances to u and v are less or equal to the distance between u and v. GG is defined such that an edge (u, v) exists if no other node w is inside the circle with the diameter uv (Fig. 7). Bose *et al.* [24] proved that the Euclidean stretch factor of GG and RNG are $\Theta(\sqrt{n})$ and $\Theta(n)$, respectively, where n is the number of points. The same construction also implies that even for a constant density point set, the topological stretch factors can be $\Omega(\sqrt{n})$ for GG and $\Theta(n)$ for RNG. If the density of the points is high, the stretch factor can be as great as $\Theta(n)$.

One problem with maintaining a sparse graph of the underlying topology is that we may have to traverse many short edges when the density of the point set is high. In GPSR, the problem is avoided by using the sparse graph only for getting out of local minimum. During the greedy geographic forwarding, the protocol considers all the visible nodes but not just adjacent nodes in the graph. Therefore, the complexity of each routing step can be high if the density is high. Nevertheless, the routing can be benefited by considering all the visible points. For example, we are able to prove the following result on the quality of path in a special case, where geographic forwarding is never stuck at a local minimum.[5]

Theorem 5.1: If a packet can be greedily forwarded from u to v, i.e., no local minimum is reached during the forwarding, then the routing path length is bounded by $O(\ell^2)$ if the shortest path between u, v has length ℓ.

Proof: Recall that by greedy forwarding, each time we check all the visible neighbors and forward the packet to the one closest to the destination. Let the path be $P_{uv} : u_1 = u, u_2, \ldots, u_m = v$. Note that the distance between u_i and v is decreasing when i increases. Since the optimum path is of length ℓ, the distance between u and v is at most ℓ. Thus, all u_i's lie in a circle of radius ℓ centered at v. Also, we know that the points u_i and u_{i+k} for $k \geq 2$ cannot see each other because otherwise we would have chosen u_{i+k} instead of u_{i+1} as the successor of u_i in the path. Therefore, the points $u_1, u_3, \ldots, u_{2\lceil m/2 \rceil - 1}$ are mutually invisible. According to a simple packing lemma, we know that there can be at most $O(\ell^2)$ such points in a disk with radius ℓ. □

Note that the preceding bound is tight. Fig. 8(a) illustrates a situation where a path with $\Theta(\ell^2)$ nodes is discovered by greedy geographic forwarding, while the optimum path has length ℓ. While considering all the visible vertices can help to skip short edges, it incurs high cost when deciding to which node the packet is forwarded. Our algorithm does not have this problem since we perform the clustering first and only maintain a sparse graph for clusterheads and gateway nodes. This effectively "smooths out" the point set so that our analysis enjoys the property that the points are distributed with constant density. In addition to the stretch factor and the low maintenance cost for RDG as described before, we can also prove the quality of perimeter routing on our graph, again in a special case. We say that a perimeter routing follows the right-hand (left-hand) rule, if we always traverse a face in a counterclockwise (clockwise) direction. We call a path connecting u, v right-sided (left-sided) if the path lies entirely on the right (left) side of the line passing through u, v. Then, we have the following.

Theorem 5.2: If the shortest path is right-sided (left-sided) and has length ℓ, then the path discovered by perimeter routing following the right-hand (left-hand) rule has length at most $O(\ell^2)$.

Proof: Suppose that the optimum path is to the right of line uv. If the perimeter routing follows the right-hand rule, then all the points traversed lie entirely inside the region bounded by the line segment uv and the optimum path from u to v [Fig. 8(b)]. The area of that region is $O(\ell^2)$. By the constant density property, the number of nodes in that region is at most $O(\ell^2)$. Therefore, the length of the path is at most $O(\ell^2)$ as well. □

The above theorem does not specify a way to figure out which side the shortest path lies. This is in general a difficult question in perimeter routing—by following a wrong direction, we may have to traverse a very long path while a short path exists by following the other direction. We do not know of any good local rule to resolve this problem. However, one trick one may use is

[5]As noted in [6], this is the typical case.

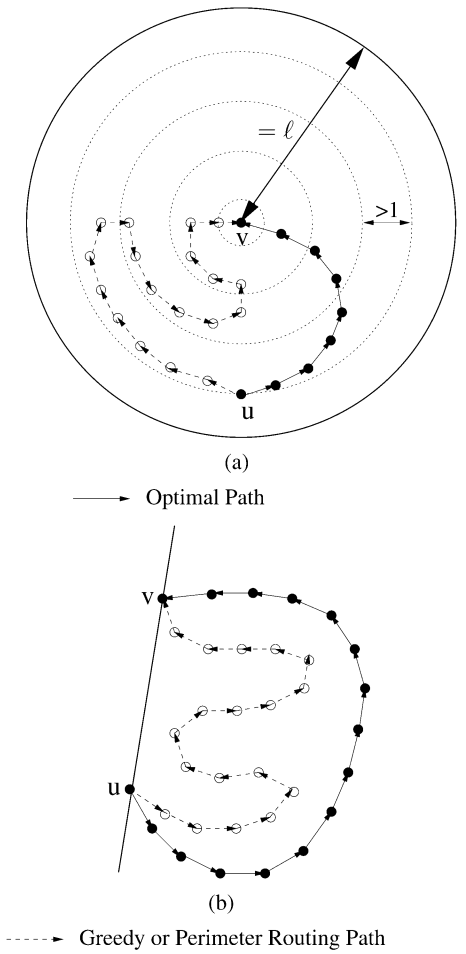


Fig. 8. Examples of greedy forwarding and one-sided perimeter routing.

to try both directions. Specifically, we can forward the packet to the right t hops, and then come back to u and forward the packet to the left t hops. Then, we double t's value and repeat the process until either we reach a point where greedy forwarding is available, or we enter another face, or we come back to the starting edge, which means there is no path between u, v. We can obtain competitive bounds by this doubling technique: if the number of nodes traversed by following the optimal direction in the perimeter routing is ℓ, then the number of nodes traversed with this scheme is $O(\ell)$.

In GPSR, when a packet cannot reach its destination, it finds this out by traversing a face boundary and returning to the edge where it began. If a destination is unreachable from the source, the undeliverable packet has to traverse the outer face of the source's connected component. The RDG shortens such travel. In the RDG, routing is done in a much smaller graph than the RNG and GG, and the undeliverable packet travels through many fewer nodes before it realizes the unreachability. This is also demonstrated by the simulation in the next section.

VI. Simulations

In the previous sections, the analyses are mostly theoretical and help us to understand the quality of the algorithm in the extreme cases. To demonstrate the quality of our algorithm in practice, we have also performed simulations on points with uniform or nonuniform distributions.

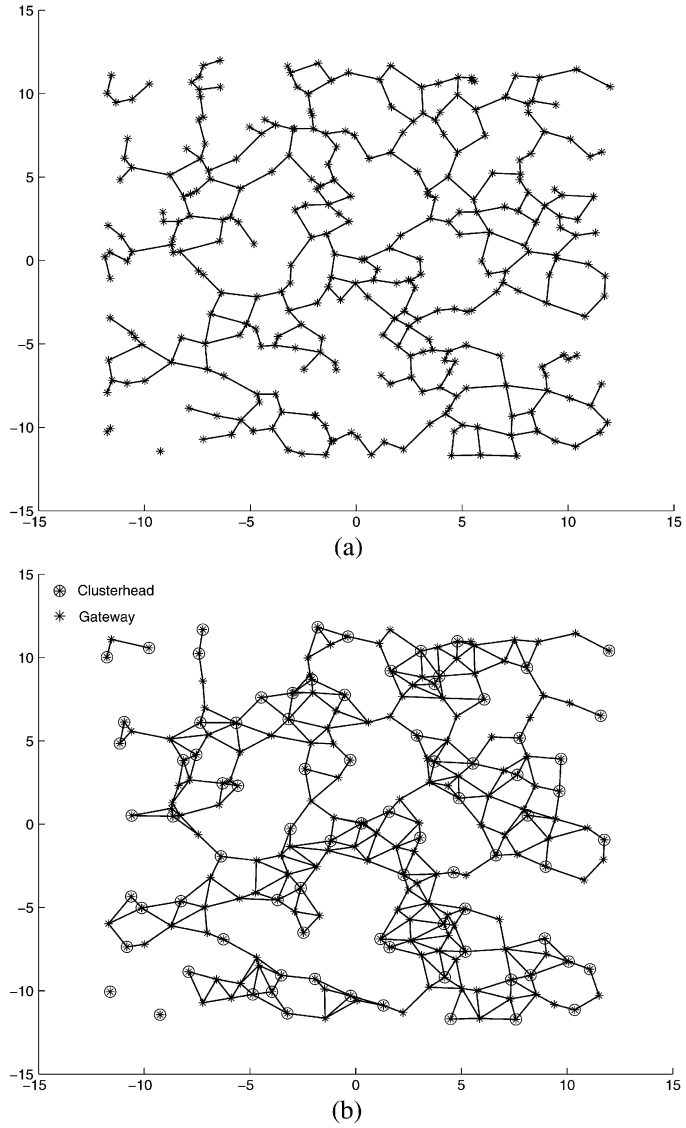


Fig. 9. (a) RNG and (b) RDG on a uniform distribution.

1) Uniform Distribution: In this simulation, we used 300 random points in a square of side length 24. Each node can see all the nodes in a disk of radius 2 around itself. The density of nodes is about 8. We only use the one-level clustering algorithm to select the clusterheads. The simulation is first done in a static case. We evaluate the quality of the path found by GPSR on the RNG and RDG. The RDG on clusterheads and gateways is shown in Fig. 9(b). The RNG is shown in Fig. 9(a) The RDG is a sparser backbone compared with the RNG, containing fewer nodes. Therefore, when we do perimeter routing along a face in the RDG, the number of hops experienced is much smaller than in the RNG. This is also shown in the simulation results. Fig. 10(a) shows the comparison of performance in the RNG and RDG. For all pairs of reachable nodes, we compute the number of hops of the optimal path, and the path we get using GPSR on both the RNG and RDG. For the pairs with the same optimal length, we take the average length of the paths from GPSR. We can see that the RDG outperforms the RNG in terms of the

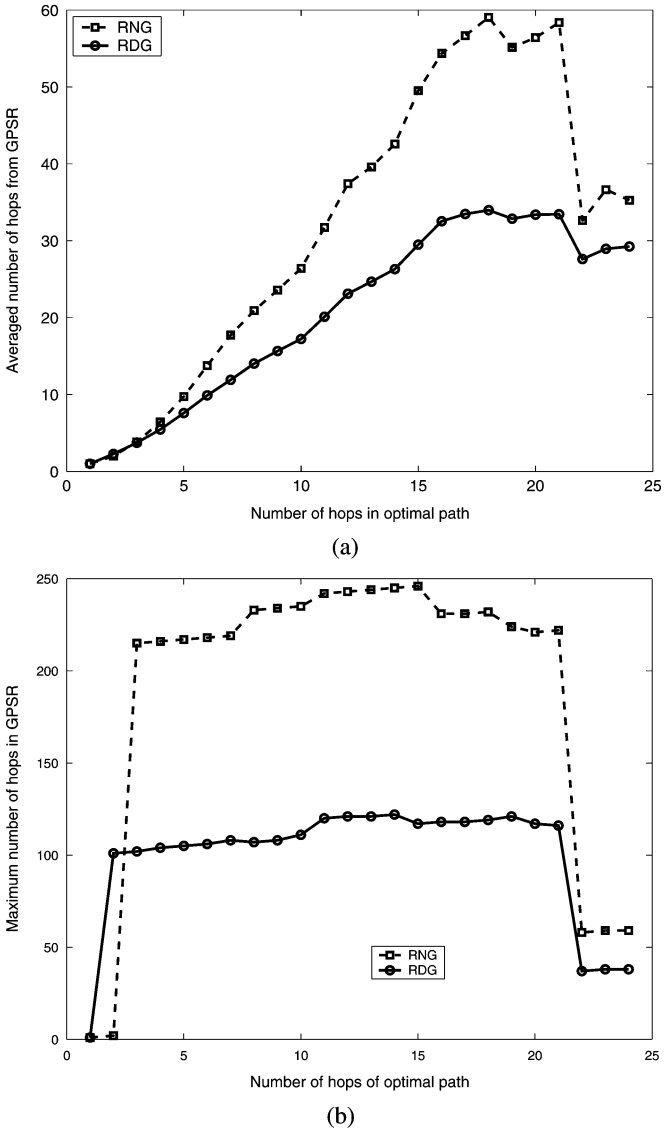


Fig. 10. (a) Averaged length using GPSR versus optimal length. (b) Maximal length using GPSR versus optimal length.

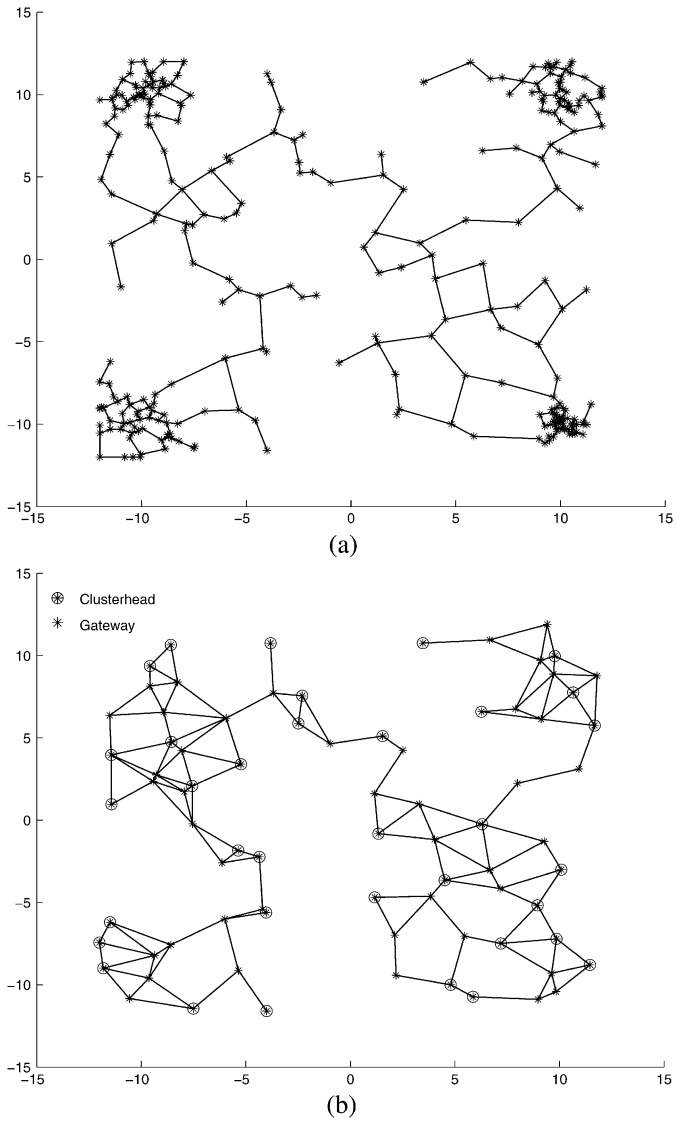


Fig. 11. (a) RNG and (b) RDG on a nonuniform distribution.

routing path quality. Fig. 10(b) shows the maximal number of hops by GPSR on RNG and RDG. Also, when we look at all the unreachable pairs, on the average 67 hops are traveled in RDG and 139 hops are traveled in RNG.

We also experimented with motion. We assume every point moves with a constant velocity in a random direction at a random speed between 0 and 1. Points are constrained to move within the square (of side length 24), so a point bounces back once hitting the virtual boundary.[6] In this paper, we are interested in how the path quality between two fixed points changes over time. We track the topology of the network under motion over 1000 frames at 1 frame/s. We then compute the path length between these two specific points at each frame. On average RDG outperforms RNG by more than 37% (23 hops versus 37 hops).

2) Nonuniform Distribution: In the real world, the nodes are far from uniformly distributed. In this case, the advantage of the RDG over the RNG is shown more obviously by the simulation. Here, we show a simulation with 300 points—100 points are randomly distributed, and another 200 points are clustered in four groups. The size of a node's visible range is a disk of radius 3.5. The RNG and RDG are shown in Fig. 11. The comparison of path length in RNG and RDG is shown in Fig. 12. We can see from the figures that most of the packets follow a shorter path in RDG, compared with RNG. The advantages are clearer when the length of the optimal path increases.

[6]This models the situation that one point moves into and one point moves out of the specified region.

VII. Discussion

We discuss some other practical issues in implementing and measuring the method in this chapter. We also state the recent results on how to find a short path in wireless ad hoc networks, to make the problem complete.

A. Scaling Versus Spanner Property

One desirable property of the routing graph is that every node has small degree, so no node can be overloaded. However, there

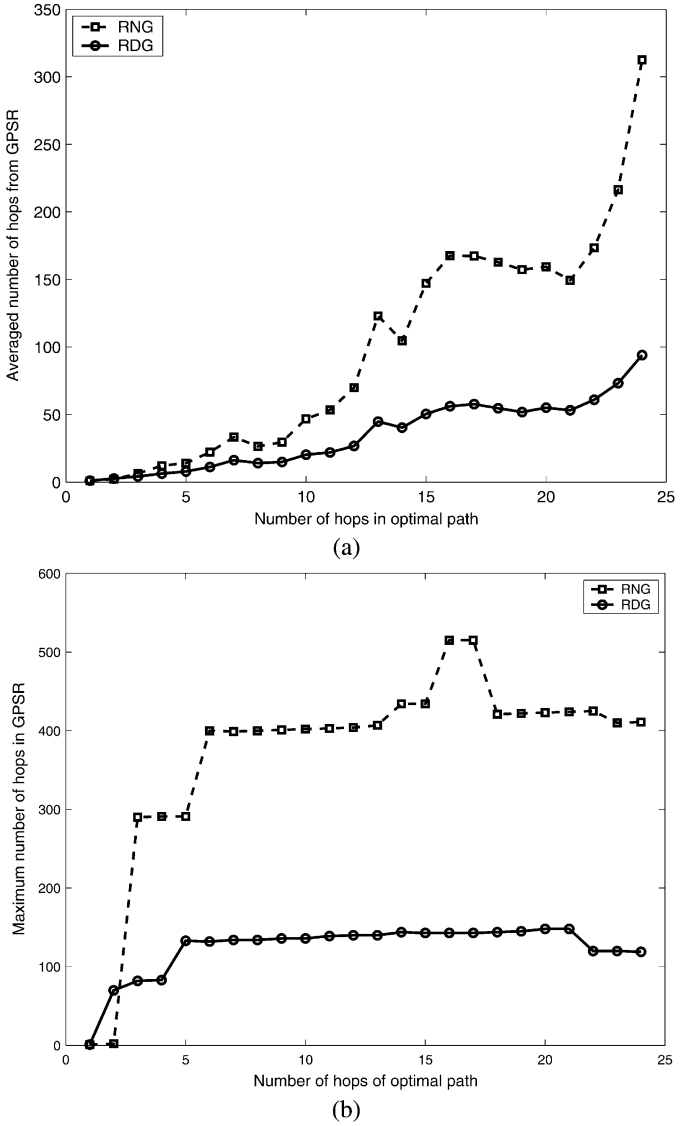


Fig. 12. (a) Averaged length using GPSR versus optimal length. (b) Maximal length using GPSR versus optimal length.

is a tradeoff between the constant degree and the spanner property. If we enforce constant degree of the routing graph, the spanner property cannot be achieved. Consider the situation in which n nodes are very near to each other such that every node can see all the other nodes. If we let each node's degree in the routing graph be at most C, then one node x can reach at most C nodes in one hop, and C^2 nodes in two hops, and C^k nodes in k hops. Then, there must exist a node that x can reach in no fewer than $\Omega(\log n)$ hops.

Notice that in our routing graph, the clusterheads may have a lot of clients, but the RDG constructed on clusterheads and gateway nodes has a constant degree due to the constant density property. So, when the messages are routed on the "backbone" graph, the routing decisions are made with $O(1)$ cost per node. This is compared with routing on RNG and GG, whose degree might be as high as $\Omega(n)$.

B. More About Clustering

One common issue in using clusterheads in routing protocols is that frequent clusterhead changes may adversely affect routing protocol performance since nodes are busy in clusterhead selection rather than packet forwarding. For example, consider the clusterhead gateway switch routing (CGSR) protocol proposed by Chiang *et al.* [11]. Each node keeps a cluster member table, where it stores the destination clusterhead for each mobile node in the network. So when a node changes its clusterhead, the updated information must be broadcast to every node in the network, which causes a lot of traffic. In addition, each node keeps a routing table that is used to determine the next hop in order to reach the destination. Changes of clusterheads also cause a lot of changes in the routing table. To minimize the changes of clusterheads, they proposed a least cluster change (LCC) clustering algorithm, in which clusterheads only change when two clusterheads come to see each other, or when a node moves out of the visible range of all the clusterheads.

However, the above is not a problem in our routing graph: GPSR does not require any routing tables. The routing graph changes locally and need not be broadcast over the whole network. In addition, our clustering algorithm is stable. Changes to clusterheads and gateways occur only when the underlying unit-disk graph changes. From [9], if all the nodes follow bounded-degree algebraic motion, the number of changes of our clustering is at most $O(n^2 \log\log n)$, which is near optimal. They also showed in [9] that under such motion, to maintain the minimum number of clusters at all times, the number of clusterhead changes is $\Theta(n^3)$. To maintain a constant c approximation, the number of clusterhead changes is at least $\Omega(n^2)$. In summary, our clustering changes only when the network topology changes. Any routing graph such as the RNG or GG needs to be updated according to the network topology as well. On the other hand, it is not the case that RDG would change more frequently than RNG or GG. Under certain conditions, RDG does not change, while both GG and RNG suffer from a lot of changes. Consider nodes moving on a line with the same speed, except a special node moves faster. Since the probability that the fast node has the ID high enough to be a clusterhead is small, most of the time the RDG does not change, but the RNG or GG could change $\Omega(n)$ times.

The theoretically proved constant approximation ratio of the clustering algorithm we used in this paper [9] is for the worst case. In practice, the approximation ratio is a small constant, as shown by the simulation. In fact, for any clustering algorithm with constant approximation ratio, the routing graph constructed in this paper will have a constant stretch factor under both topological and Euclidean distance measures. The clustering algorithm proposed by Hershberger [16] gives a nine-approximation ratio in the worst case. The clusters can also be maintained efficiently under motion., but the algorithm requires global information and is, thus, less well suited for the distributed environment. The greedy algorithm, i.e., each time an uncovered node claims itself as a clusterhead until all nodes are covered and no clusterheads are within distance 1 of each other, has an approximation ratio 6 by a simple packing argument. This algorithm can be implemented under motion in a distributed fashion: when two clusterheads are too close to each other, one of them retires and the uncovered nodes claim themselves as clusterheads by using a random backoff scheme (additional inconsistency resolution mechanism is necessary here); when one node moves outside its clusterhead's range, it either finds another clusterhead or claims

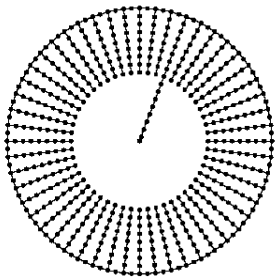


Fig. 13. Lower bound construction for the online localized routing problem from [25]. There are multiple spikes on a circle pointing inside. Only one of them connects to the destination, which lies at the center of the circle. Any local algorithm has no idea which spike will lead to the destination and has to try all of them in the worst case.

itself as a clusterhead if it fails in finding one. The problem with this algorithm is that it suffers from expensive events. When a clusterhead retires, suddenly there could be $\Omega(n)$ nodes looking for new clusterheads. Again, we emphasize here that users can choose their favorite clustering algorithms suitable for their applications and the routing graph is a spanner as long as the clustering algorithm has an $O(1)$ approximation ratio.

C. Finding a Short(er) Path

This paper proposed a spanner for ad hoc wireless networks that *contains* a path whose length is within a constant factor of the shortest-path length. To make the solution complete, the natural follow-up question is how to find this path. In Section V, we show that geographical routing under certain circumstances produces paths with length $\Theta(k^2)$, if the shortest-path length is k. The results for finding a constant approximate routing path are listed below.

If only local information is available, i.e., each node knows only the nodes within a constant number of hops, then Kuhn *et al.* [25] showed a lower bound construction in which any online algorithm finds a path of length $\Omega(k^2)$ if the shortest path has length k, as shown in Fig. 13. They also proposed a geometric routing algorithm that achieves the $O(k^2)$ bound.

On the other hand, if the topology of the whole network is available, Gao and Zhang [26] proposed an algorithm that preprocesses the unit-disk graph into a structure of size $O(n \log n/\varepsilon^4)$ such that any $(1+\varepsilon)$-approximate shortest distance query is answered in $O(1)$ time, for any $\varepsilon > 0$. The $(1+\varepsilon)$-approximate shortest path can be output in $O(k+\log n)$ time, where k is the length of the shortest path.

D. Disadvantages of Geographical Routing

This paper focuses on constructing good routing graphs for geographical routing. The most attractive properties of geographical routing are its efficiency, locality, and scalability. We should also note the disadvantages of geographical routing. Obtaining the location information is difficult or expensive. GPS receivers can be costly and do not work indoors, or in conditions under heavy foliage. Various localization algorithms that estimate locations by the distance to certain anchor nodes were proposed, but they are still computation intensive and the accuracy is not satisfactory. Furthermore, updating and querying location information adds nonnegligible cost to the geographical routing paradigm, especially in highly mobile networks. Designing robust, universal, and scalable routing protocols still remains an open and important problem in wireless ad hoc networks.

VIII. Summary and Future Work

In this paper, we have presented a maintainable routing graph R that is a planar spanner of the full connectivity graph. We have assumed that all nodes have equal and circular communications ranges. In practice, this assumption is often violated due to fading and multipath effects. Even when equal and circular communication ranges are possible, there may be energy advantages to using shorter ranges in areas of higher node density, so as to conserve energy. It would be interesting to extend the results of this paper to these more realistic scenarios.

References

[1] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Pers. Commun.*, vol. 6, no. 2, pp. 46–55, Apr. 1999.

[2] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in *ad hoc* networks of sensors," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom 2001)*, Rome, Italy, Jul. 2001, pp. 166–179.

[3] A. Savvides and M. B. Strivastava, "Distributed fine-grained localization in *ad-hoc* networks," *IEEE Trans. Mobile Comput.*, submitted for publication.

[4] J. Li, J. Jannotti, D. Decouto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *Proc. 6th ACM/IEEE Int. Conf. Mobile Comput. Netw.*, 2000, pp. 120–130.

[5] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Pers. Commun.*, vol. 8, no. 1, pp. 48–57, Feb. 2001.

[6] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. o ACM/IEEE Int. Conf. Mobile Comput. Netw.*, 2000, pp. 243–254.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Netw.*, vol. 7, no. 6, pp. 609–616, 2001.

[8] *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, Eds., North-Holland, Amsterdam, The Netherlands, 2000, pp. 425–461. D. Eppstein, "Spanning trees and spanners".

[9] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Discrete mobile centers," *Discrete Comput. Geometry*, vol. 30, no. 1, pp. 45–65, 2003.

[10] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-min D-cluster formation in wireless ad hoc networks," in *Proc. 19th IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 1999, pp. 32–41.

[11] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," in *Proc. IEEE SICON*, Apr. 1997, pp. 197–211.

[12] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," in *Proc. IEEE*, vol. 75, Jan. 1987, pp. 56–73.

[13] M. Gerla and J. Tsai, "Multicluster, mobile, multimedia radio network," *ACM-Baltzer J. Wireless Netw.*, vol. 1, no. 3, pp. 255–265, 1995.

[14] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. IEEE Int. Conf. Commun. (ICC'97)*, Jun. 1997, pp. 376–380.

[15] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proc. 3rd Int. Workshop Discrete Algorithms Methods Mobile Comput. Commun.*, Aug. 1999, pp. 7–14.

[16] J. Hershberger, "Smooth kinetic maintenance of clusters," in *Proc. ACM Symp. Comput. Geometry*, Jun. 2003, pp. 48–57.

[17] L. P. Chew, "There is a planar graph almost as good as the complete graph," in *Proc. 2nd Annu. ACM Symp. Comput. Geometry*, Jun. 1986, pp. 169–177.

[18] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay graphs are almost as good as complete graphs," *Discrete Comput. Geometry*, vol. 5, no. 4, pp. 399–407, 1990.

[19] J. M. Keil and C. A. Gutwin, "The Delaunay triangulation closely approximates the complete Euclidean graph," in *Lecture Notes in Computer Science*, ser. , 1989, vol. 382, Proc. Int. Workshop Algorithms Data Structures, pp. 47–56.
[20] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanner for routing in mobile networks," in *Proc. 2nd ACM Symp. Mobile Ad Hoc Netw. Compu.*, Oct. 2001, pp. 45–55.
[21] X.-Y. Li, G. Calinescu, and P.-J. Wan, "Distributed construction of planar spanner and routing for ad hoc networks," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1268–1277.
[22] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Fieder, "Geometric spanners for wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 5, May 2003.
[23] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.
[24] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, "On the spanning ratio of Gabriel graphs and β-skeletons," in *Proc. 5th Latin Amer. Symp. Theoretical Inf.*, Apr. 2002, pp. 479–493, submitted for publication.
[25] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Asymptotically optimal geometric mobile ad-hoc routing," in *Proc. 6th Int. Workshop Discrete Algorithms Methods Mobile Comput. Commun.*, Sep. 2002, pp. 24–33.
[26] J. Gao and L. Zhang, "Well-separated pair decomposition for the unit-disk graph metric and its applications," in *Proc. 35th ACM Symp. Theory Comput.*, Jun. 2003, pp. 483–492.

Jie Gao (M'01) received the B.S. degree from the University of Science and Technology of China in 1999 and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 2004.

She is currently a Postdoctoral Fellow at the Center for the Mathematics of Information, California Institute of Technology, Pasadena. She will join the State University of New York, Stony Brook, as an Assistant Professor in Fall 2005. Her research interests are algorithms, ad hoc communication and sensor networks, and computational geometry.

Leonidas J. Guibas received the Ph.D. degree from Stanford University, Stanford, CA, in 1976.

He heads the Geometric Computation Group in the Computer Science Department, Stanford University, where he works on algorithms for sensing, modeling, reasoning, rendering, and acting on the physical world. He has published and lectured extensively in computational geometry, geometric modeling, computer graphics, computer vision, robotics, and discrete algorithms. His main subsequent employers were Xerox PARC, MIT, and DEC/SRC. He has been with Stanford University since 1984 as a Professor of Computer Science.

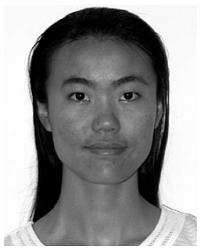
Prof. Guibas is an ACM Fellow.

John Hershberger received the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1987.

He joined the DEC Systems Research Center in 1987 and moved to Mentor Graphics Corporation, Wilsonville, OR, in 1993. He holds a courtesy appointment in the Department of Computer Science, University of California, Santa Barbara. His research interests include computational geometry, analysis of algorithms and data structures, and electronic design automation.

Li Zhang received the B.E. and M.E. degrees, both in computer science, from Tsinghua University, Beijing, China, in 1991 and 1993, respectively, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 2000.

From 1993 to 1995, he was an Assistant Teacher in the Department of Computer Science, Tsinghua University. In 2000, he joined Compaq (formerly DEC) Systems Research Center, which became a part of Hewlett Packard Laboratories, Palo Alto, CA, in 2002. He is currently a Researcher at the Information Dynamics Laboratory, Hewlett Packard Laboratories. His research interests include computational geometry, algorithms for ad hoc and sensor networks, and game theoretical analysis of resource allocation.



An Zhu received the B.S. and M.S. degrees in computer science and mathematics from the University of Maryland, College Park, in 1999. She is currently working towards the Ph.D. degree in computer science at Stanford University, Stanford, CA.

Her research interests are design and analysis of algorithms, with applications to scheduling, networks, database systems, etc.