Geometric Spanner for Routing in Mobile Networks

Jie Gao* Leonidas J. Guibas* John Hershberger† Li Zhang‡ An Zhu*

Abstract

We propose a new routing graph, the Restricted Delaunay Graph (RDG), for ad hoc networks. Combined with a node clustering algorithm, RDG can be used as an underlying graph for geographic routing protocols. This graph has the following attractive properties: (1) it is a planar graph; (2) between any two nodes there exists a path in the RDG whose length, whether measure in terms of topological or Euclidean distance, is only a constant times the optimum 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 also show by simulation that the RDG outperforms the previously proposed routing graphs under the Greedy Perimeter Stateless Routing (GPSR) protocol. In addition, we investigate theoretical bounds on the quality of paths discovered using GPSR.

1 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. Each node's transmission range is usually modelled as a fixed-radius circle centered at the node (we assume in this paper that all nodes have the same transmission range). In an ad hoc network, there is no centralized control or fixed infrastructure. Each mobile node operates as a router, relaying packets for other nodes that may not be within direct transmission range (there might be some nodes that cannot work as routers, but we will not consider them in this paper). When the nodes move, the underlying topology of the network changes constantly.

The constantly changing network topology makes routing in ad hoc networks difficult. Many routing protocols have been proposed for mobile networks. In on-demand routing [17, 18, 21], a node floods the network to discover a path when it initiates communication to another node and then caches the path for later use. However, flooding is expensive, and the lifetime of a cached path is very short if the nodes move quickly. One way to avoid the expense of full flooding is for each node to communicate only with a selected subset of neighboring nodes [13, 20, 22]. This approach can be viewed as maintaining a subgraph of the underlying topology according to certain local rules. The subgraph should at least preserve the connectivity, i.e., any two nodes that are connected in the underlying graph should be connected in the subgraph as well.

Another way to avoid flooding is to use *geographic forwarding*, which is applicable if the geographic location of each node is available to other nodes from a location service [2, 15, 16]. 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, repeating the process until the packet reaches the destination. 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 may get 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. To deal with this problem, Karp and Kung proposed the Greedy Perimeter Stateless Routing (GPSR) protocol, which guarantees the delivery of the packet if a path exists [13]. Their method requires the maintenance of a planar subgraph of the underlying connectivity. When a packet is stuck at a node, the protocol will route the packet around the faces of the graph to get out of the local minimum. Karp and Kung proposed to use two planar subgraphs: the *relative neighborhood graph* (RNG) and the *Gabriel graph* (GG), on the original node set. While the algorithm performs quite well when each individual node's visible range is large and nodes are randomly distributed, it does not perform well in more general cases. 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 [10]. This fact limits the quality of paths even if we use globally optimum routing methods on these subgraphs.

The *stretch factor* of paths in a graph captures this aspect of path quality. Roughly speaking, the stretch factor of a

*Department of Computer Science, Stanford University, Stanford, CA 94305. E-mail: jgao,guibas,anzhu@cs.stanford.edu.

†Mentor Graphics, 8005 S.W. Boeckman Road, Wilsonville, OR 97070. E-mail: john_hershberger@mentor.com.

‡Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301. E-mail: l.zhang@compaq.com.

subgraph G' of a graph G measures the worst-case ratio between the length of a shortest path in G' to the length of the shortest path with the same endpoints in G.

We present a routing graph, the *restricted Delaunay graph* (RDG), that has nice theoretical guarantees on the stretch factor of the paths. In particular, the RDG has paths with Euclidean or topological length only a constant factor longer than the length of the optimal path. Our routing graph can be maintained efficiently in a distributed fashion under node motion. In addition to presenting rigorous theoretical analysis, we also demonstrate that experimentally, GPSR on our routing graph improves the routing path quality compared to the path quality using the GG or RNG, under both uniform and clustered distributions of the points.

To define our graph, we first group points 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 non-neighbor node v, u first forwards the packet to its clusterhead, and the packet is then forwarded among 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 [11]. This simplifies forwarding during routing. In [13], the greedy geographic forwarding is done by examining all the neighboring nodes in order to skip short edges in the graph. This process is expensive when the nodes are densely distributed. In our routing graph, since we cluster nodes in the first stage, we can perform the greedy geographic forwarding by considering only the adjacent nodes in the *routing graph*, and this reduces the complexity significantly. The clustering algorithm also improves the behavior of GPSR, since we have only essential points in the graph. In the GG or RNG, GPSR may traverse a short boundary that consists of a dense sequence of points; but boundaries in the RDG have only constant density. We also investigate the trade-off between scaling and the spanning property, and the efficiency of clusterhead changes.

The rest of the paper is organized as follows: Section 2 surveys previous work, Section 3 gives a detailed description of the RDG and proves the spanning property, Section 4 deals with the distributed implementation of the RDG, Section 5 proves theoretical bounds on the length of the actual routing paths under certain circumstances, Section 6 compares the simulation results of GPSR on the RDG vs. GPSR on the RNG and discusses various aspects of the RDG; finally, Section 7 summarizes our work and gives some future directions for research.

2 Previous Work

Many routing protocols have been developed for ad hoc mobile networks. Most of them fall into two categories, table-driven and source-initiated on-demand. A review of current routing protocols can be found in [21]. Recently, Karp and Kung proposed the Greedy Perimeter Stateless Routing (GPSR) protocol, in which only the positions of routers and a packet's destination are used to make packet forwarding decisions. GPSR makes greedy forwarding decisions using only information about a router's immediate neighbors in the network topology. When greedy forwarding is not possible, the packet is forwarded along the perimeter of a face in a planar graph. The advantage of GPSR over other routing protocols is that forwarding decisions are made using local information; there is no need to maintain routing tables or make global broadcasts. This idea of combining greedy and perimeter routing on planar graphs is independently investigated by Bose et al. [4].

Clustering is used in many routing protocols in mobile networks [1, 6, 9, 12]. Our basic clustering algorithm is similar to the Lowest-ID Cluster Algorithm proposed by Ephremides, Wieselthier, and Baker [9, 12]. A similar idea also led to the Max-Min D-clustering scheme proposed by Amis et al. [1]. Chiang et al. [6] proposed a Least Cluster Change clustering (LCC) algorithm that tries to minimize clusterhead changes. 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. Another class of routing protocols is based on the Minimum Connected Dominating Set (MCDS) [7, 23]. Das et al. [7] 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) [23].

Spanner graphs have been heavily studied in computational geometry [10]. The Delaunay triangulation has been shown to be a planar spanner [5, 8, 14]. However, little is known about restricted spanner graphs, where only edges shorter than 1 are allowed. Also, little is known about how to maintain spanner graphs in a distributed manner when the points move.

3 Routing Graph with Constant Stretch Factor

We assume that two mobile nodes can communicate to each other directly if their separation is no larger than 1. We call two such nodes *visible* to each other. The visibility 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 might 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 (Euclidean length) 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 subgraph G with constant stretch factor. This graph G can serve as a routing graph in the ad hoc network.

Our construction consists of two phases. First, we make use of the hierarchical clustering algorithm in [11] to select a small subset of V, called *clusterheads*, so that each node in V can communicate directly to some clusterheads. Each non-clusterhead 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 the routing in $\mathcal{G}$ to routing between clusterheads and gateways. Second, we form a planar routing graph on the clusterheads and gateways by applying a local rule, called the *restricted Delaunay edge* rule. The graph produced this way is called the *restricted Delaunay graph* (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:

- RDG is a planar graph (no two edges cross each other in the graph).
- R has constant stretch factor, in both topological and Euclidean senses. That is, 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.
- R can be efficiently computed and maintained in the distributed setting.

In the rest of this section, we will first briefly describe the clustering algorithm in [11] and then present the restricted Delaunay graph.

3.1 Mobile clustering

The goal of clustering is to select a subset of nodes as the *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 [11] is used because we need some special properties of the clustering algorithm to achieve good properties on the routing graph. The one-level clustering algorithm analyzed in [11] works as follows: assume a random ordering on the IDs of the nodes, and let each node nominate the node with the highest ordered ID in its visible range [1]. 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, Wieselthier, and Baker [9], but without theoretical analysis. In [11], 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. The overall outline of the algorithm is as follows: Initially at round 0, all nodes are clusterheads and participants. At each round each 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 $2^i/\log n$. 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 [11]. The following result is from [11].

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

In the rest of the paper, we assume the clusterheads have constant density, with the understanding that this is in the expected sense.

To enable different clusters to communicate with each other, we introduce *gateways* [9]. 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 it is nominated by others to be a clusterhead. We allow such situation. A 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 points in Section 4. From Theorem 3.1, we can also derive the following fact. The proof is omitted in this abstract.

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

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

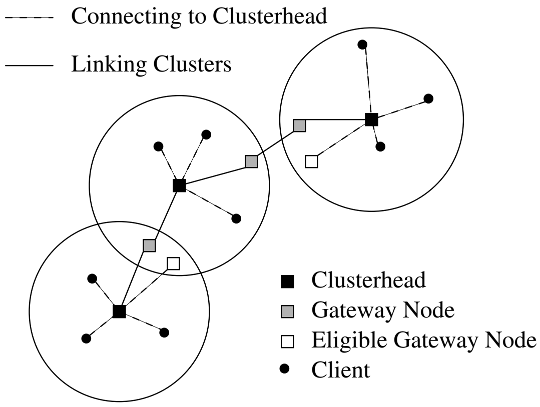


Figure 1: Example of Linked Cluster organization of a mobile network

The hierarchical algorithm provides a theoretical bound that holds for *any* distribution of points in space. In reality, distributions that cause bad clustering quality appear very rarely. In our simulation, we only use the one-level clustering algorithm described above.

3.2 Restricted Delaunay graph

Our routing graph R includes the edges that connect each client to the clusterhead it nominated. In addition, we connect together clusterheads via gateways to create a planar graph among clusterheads and gateways. Figure 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 a *restricted Delaunay graph* (RDG) for connecting clusterheads and gateways, similar to the GG and RNG used for all the nodes in the perimeter routing paper [13]. The difference is that the RDG provides theoretical guarantees on the Euclidean and topological stretch factors, while the GG and RNG do not (Section 5). In the rest of this section, we concentrate on the visibility 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.

3.2.1 Voronoi diagram, Delaunay triangulation and Restricted Delaunay graph

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. These classical geometric structures have numerous applications [19]. 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. Figure 2 shows an example of a Voronoi diagram and Delaunay triangulation of a set of points.

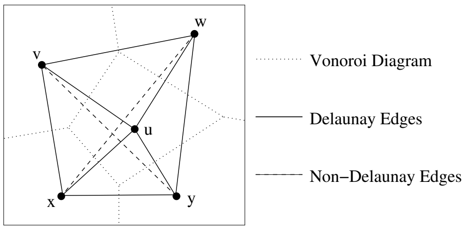


Figure 2: Voronoi diagram and Delaunay triangulation of a set of points.

The Delaunay triangulation is known to be a good spanner of the complete graph [8, 14]. However, we cannot use this graph directly in our setting because (1) the Delaunay triangulation may have very long edges, while we are only allowed to connect points within distance 1; and (2) the empty-circle rule is a global rule that is not suitable for local computation. To deal with those two problems, we define the *restricted Delaunay graph* (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 D-edges).*

Notice that the restricted Delaunay graph always exists and is not necessarily unique. By its planarity, we also know that RDG is sparse, i.e. has linearly many edges in terms of the number of nodes. In the following, we will first show that any RDG has nice spanning properties. In Section 4, we will show how to maintain RDG, and therefore our routing graph R, for mobile points.

3.2.2 Spanning properties of R

The Euclidean spanning property of the Delaunay graph was first proved in [8] and later improved in [14]. We extend the proof in [8] to show the Euclidean distance spanning property of the graph S, which is a subgraph of a visibility graph $\overline{G} = (\overline{V}, \overline{E})$ and contains only the short D-edges. Then the RDG graph is also an Euclidean distance spanner, since it contains all the short D-edges.

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

Proof: It suffices to prove that for any two nodes $u, v \in \overline{V}$, 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 proven for regular Delaunay triangulations by Dobkin et al. [8]: 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 $\frac{1+\sqrt{5}}{2}\pi \cdot \ell$. 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 [8] are short D-edges, which all exist in S. $\square$

While the above lemma shows that RDGs are good Euclidean spanners, a 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 length of the shortest path is $\Theta(n)$. However, if the 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 the topological stretch factor is bounded. Fortunately, the graph G_{CG} has constant density by Corollary 3.2.

Lemma 3.5 *A RDG is a topological spanner graph with constant stretch factor. That is, for any two nodes u, v in G_{CG}, $\tau_{RDG}(u, v) \leq C_2 \cdot \tau(u, v)$ 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. $\square$

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

Theorem 3.6 *Graph R is a both Euclidean and topological spanner graph with constant stretch factor.*

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

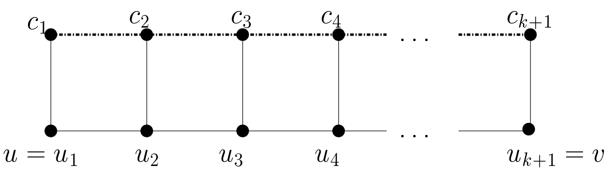


Figure 3: Spanner property of routing graph R

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., $\tau_{G_{CG}}(c_i, c_{i+1}) \leq 3$ (Figure 3). From lemma 3.5, there exists a path P_i in the RDG whose length is at most $C_2 \cdot \tau_{G_{CG}}(c_i, c_{i+1})$. 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 $\tau_R(u, v) \leq 2 + \sum_{i=1}^{k} C_2 \cdot \tau_{G_{CG}}(c_i, c_{i+1}) \leq 3C_2 \cdot k + 2$.

The Euclidean spanner property follows from the constant density of the clusterheads and gateways. Basically all the c_i lies 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. $\square$

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 also prove some theoretical bounds on the length of the actual paths used by geographic forwarding. In addition, our experimental results show that our routing graph performs better than the RNG or GG under geographic forwarding (Section 6).

4 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 efficiently and consistently maintain the relevant part of the routing graph, with only the knowledge of the neighbors. For the maintenance of clusterheads, we use the algorithm described in [11] and refer the reader to that paper for details. Here we will describe the maintenance of restricted Delaunay graphs and gateway nodes.

4.1 Maintaining RDG

According to Lemma 3.5, any RDG has the desired spanning property. We will describe a distributed algorithm which maintains a RDG at any instance of time. At the end of the algorithm, 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 neighbors $N(u)$ and computes the Delaunay triangulation, denoted by $T(u)$, on $N(u)$. Since $T(u)$ is computed only on $N(u)$, the edges we obtained is a superset of the short Delaunay edges, and some of them might be non-Delaunay edges. Further, the local graphs 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, we perform one round of information propagation. In the second step, each node u sends $T(u)$ to all of its neighboring nodes. Then each node executes the pseudo-code shown in Figure 4.

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

Figure 4: Pseudo-code for resolving inconsistency

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 pseudo-code 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 present in all the local graph $T(w)$ for $w \in N(u) \cap N(v)$. Therefore, the properties (1),(2), and (4) hold. The following simple geometric fact shows that property (3) holds as well.

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 $\leq$ 1) intersect at point p. By 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 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 the four nodes. Therefore, w must have received $T(u)$ when computing $E(w)$. According to Figure 4, both uv and wx must present in $T(u)$, contradicting that $T(u)$ is a planar graph. The above procedure 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 constant time for each u.

4.2 Maintaining gateway nodes

The maintenance of gateway nodes is similar to the method described in [9]. Essentially, every node broadcasts its entire neighbor set to every neighboring node. However, each node might take $O(D^2)$ time processing and making decision about whether or not it should serve as a gateway (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 make the decision. Note that changes to clusterheads and gateways occur only if the underlying visibility 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 nearby nodes of u and v need proper update, and the update time is constant per node.

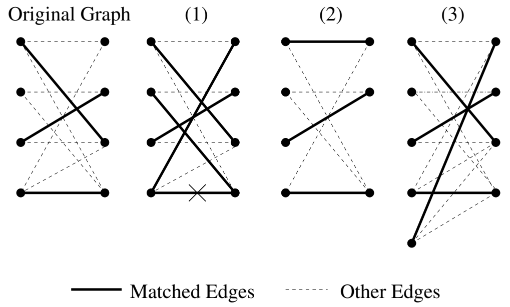


Figure 5: Maximal matching in bipartite graph $B(c_1, c_2)$. Left: original graph. (1)a pair of nodes become invisible. (2) a node leaves the cluster. (3)a new node joins the cluster

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. Figure 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 selects the matching and informs 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.

We now turn to the maintenance of a maximal matching. We first describe how the neighborhood information is organized inside a node u. To be more specific, each node v would propagate information by broadcasting an update

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
	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

Figure 6: Organizing neighbors

entry of the following form | ID | c | c' | c_1 | c_2 | ... | , where the ID uniquely identifies v, c and c' are the ID's of the clusterheads of the clusters v belongs to(recall that a node may belong to two clusters) , $c_1, c_2, \ldots$ are the ID's of 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 ID's (Figure 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 doesn't have to include all pairs of clusterheads, rather only clusterheads that are nearby. 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 to 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_2 and c_1. 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 (Figure 5(1)). 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 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 (Figure 5(2)). When p appears in $C(c_1)$, p needs to find among its neighbors nodes that belong to other some 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(Figure 5(3)).

5 Quality Analysis of Routing Graphs

As shown in Section 3.2.2, our routing graph has bounded geometric and topological stretch factors. In 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 non-planar.

Relative Neighborhood Graph (RNG) and Gabriel Graph (GG) are the planar graphs used in [13]. 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 (Figure 7). Bose et al. [3] 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 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 $\Theta(n)$.

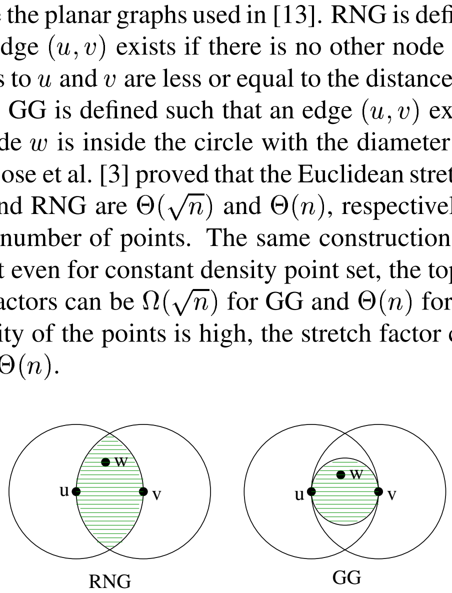


Figure 7: RNG and GG

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

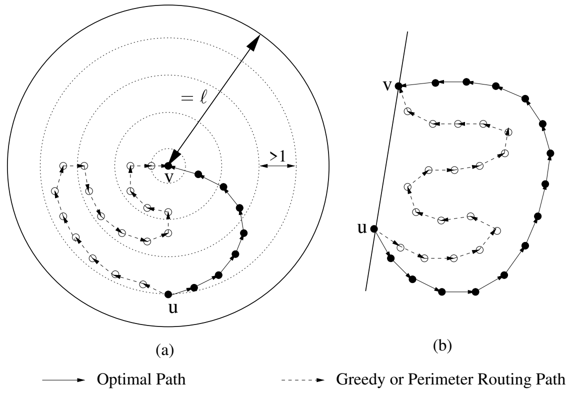


Figure 8: Examples of Greedy Forwarding and One-sided Perimeter Routing

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 the forwarding has never stuck at a local minimal[2].

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 above bound is tight. Figure 5 (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 sparse graph for clusterheads and gateway nodes. This effectively "smooth 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 call that a perimeter routing follows right-hand (left-hand) rule, if we always traverse a face in a counter-clockwise (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 that

Theorem 5.2 *If the shortest path is right-sided (left-sided) and has length ℓ, then the path discovered by the perimeter routing following right (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 (Figure 5 (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 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 may obtain competitive bounds about this method, but the details are omitted in this paper.

In GPSR, when a packet cannot reach its destination, it would find this out by traversing along a face and getting back to the edge where it began. If the destinations are beyond the source's reachable capacity, the undeliverable packets have to travel at 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 much fewer nodes before it realizes the unreachability. This is also demonstrated by the simulation in the next section.

6 Experiments and Discussions

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, we have also performed simulations on various distribution of points. We will summarize those experimental results in

[2] As noted in [13], this is the typical case.

this section and discuss some other practical issues in implementing and measuring the method.

6.1 Simulation results

We conduct experiments under both uniform and non-uniform distributions.

6.1.1 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 firstly 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 as Figure 9(b). The RNG is as Figure 9(a).

The RDG is a sparser backbone compared to the RNG, containing fewer nodes. Therefore, when we do perimeter routing along a face in RDG, the number of hops experienced is much smaller than in the RNG. This is also shown in the simulation results. Figure 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 routing path quality. Figure 10(b) shows the maximal number of hops by GPSR on RNG and RDG. Also, when we look at all the unreachable pairs, on average 67 hops are travelled in RDG and 139 hops are travelled in RNG.

We also experiment 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 [3]. In this study we are interested in how the path quality between 2 fixed points changes over time. We track the topology of the network under motion over 1000 frames at 1 frame per second. We then compute the path length between these two specific points per each frame. On average RDG outperforms RNG by more than 37% (23 hops vs. 37 hops).

6.1.2 Non-uniform 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 Figure 11.

[3]this models the situation that one point moves into and one point moves out of the specified region

The comparison of path length in RNG and RDG is shown as Figure 12. We can see from the figures that most of the packets follow a shorter path in RDG, compared to RNG. The advantages are clearer when the length of the optimal path gets longer.

6.2 Discussion

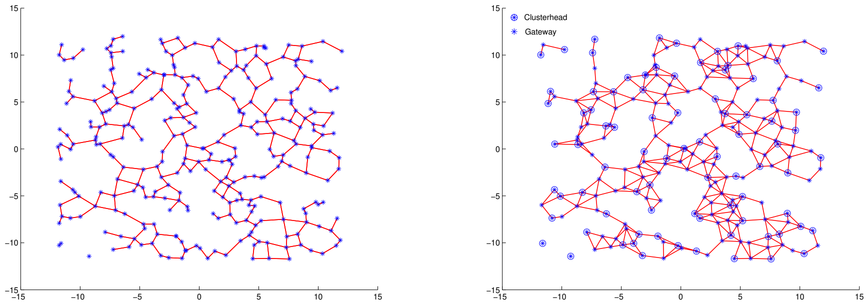
6.2.1 Scaling vs. spanner property

Another desirable property of the routing graph is that every node has small degree, so no node can be overloaded. However, there is a trade-off between the constant degree and the spanner property. If we allow constant degree of the routing graph, the spanner property can't 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 at least $\log n$ hops.

6.2.2 Efficiency of clusterheads

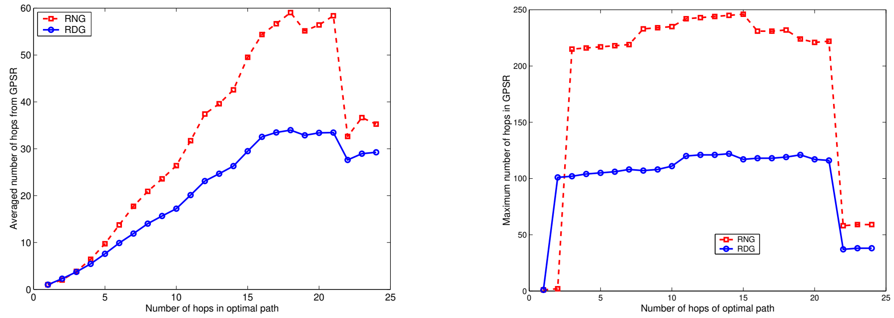
One common issue in using clusterheads in routing protocols is that, frequent clusterhead changes may adversely affect routing protocol performance since nodes are busy in clusterhead selection rather than packet forwarding. For example, consider the Clusterhead Gateway Switch Routing (CGSR) protocol proposed by Chiang et al. [6]. 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 the 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 doesn't require any routing tables. The routing graph changes locally and needs not be broadcast over the whole network. Changes to clusterheads and gateways occur only when the underlying visibility graph changes. In addition, from [11], 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 [11] 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



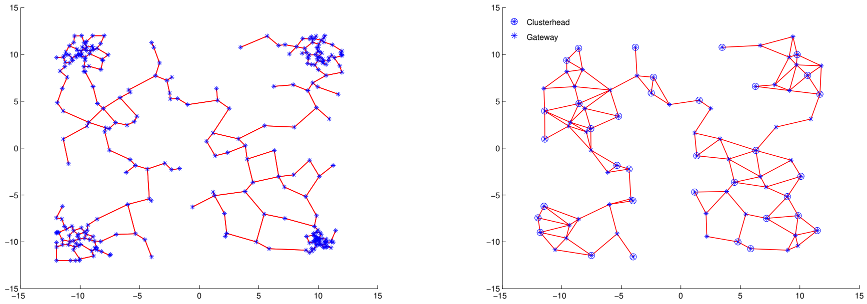
(a) (b)

Figure 9: (a) RNG (b) RDG on a uniform distribution



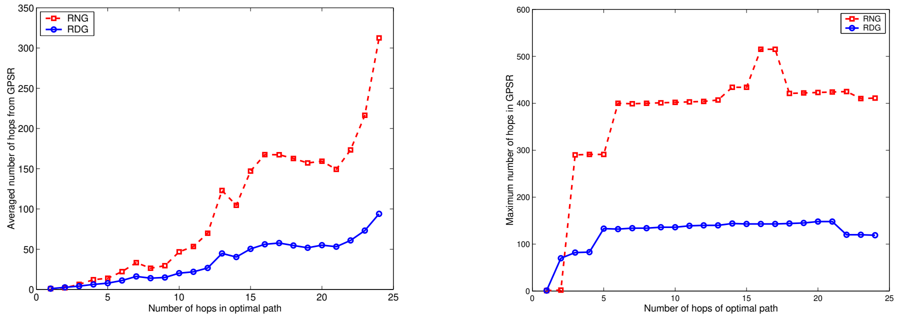
(a) (b)

Figure 10: (a) Averaged length using GPSR vs. optimal length (b) Maximal length using GPSR vs. optimal length



(a) (b)

Figure 11: (a) RNG (b) RDG on a non-unform distribution.



(a) (b)

Figure 12: (a) Averaged length using GPSR vs. optimal length (b) Maximal length using GPSR vs. optimal length

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 doesn't 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 doesn't change. But the RNG or GG could change $\Omega(n)$ times.

7 Summary and Future Work

In this paper, we have presented a routing graph that is a planar spanner. In this work we assumed that pathloss between two nodes in a wireless network is entirely dependent on the distance between those nodes. We would like to further investigate the possibility of incorporating other issues such as terrain and interference between nodes. Another interesting open problem is how to find a short path in this routing graph. Currently we use GPSR protocol on our routing graph. But only a $O(k)$ approximation bound is given for greedy forwarding and perimeter routing in the right direction. It would be good to design a routing protocol on the RDG that can find a path within a constant approximation of the optimal path.

8 Acknowledgement

The authors wish to thank Philippe Golle and Petros Maniatis for reading a draft of this paper.

References

[1] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-Min D-cluster formation in wireless ad hoc networks. In *19th IEEE INFOCOM*, March 1999.

[2] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proc. ACM/IEEE MobiCom*, pages 76–84, 1998.

[3] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and β-skeletons. *submitted to SIAM Journal on Discrete Mathematics*, 2001.

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

[5] L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 169–177, 1986.

[6] 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 '97*, pages 197–211, Apr 1997.

[7] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications (ICC'97)*, 1997.

[8] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 20–26, 1987.

[9] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proc. of IEEE*, 75(1):56–73, Jan 1987.

[10] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[11] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. 17th ACM Symp. on Computational Geometry*, Jun 2001.

[12] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM-Baltzer Journal of Wireless Networks*, 1(3), 1995.

[13] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[14] J. M. Keil and C. A. Gutwin. The delaunay triangulation closely approximates the complete euclidean graph. In *Proc. International Workshop on Algorithms and Data Structures*, volume 382 of *Lecture Notes Comput. Sci.*, pages 47–56, 1989.

[15] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proc. ACM/IEEE MobiCom*, pages 66–75, 1998.

[16] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proc. 6th Annu. ACM/IEEE International Conference on Mobile Computeing and Networking.*, 2000.

[17] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. IEEE Infocom*, pages 1405–1413, 1997.

[18] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[19] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[20] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE J. Selected Areas in Communications*, 17(8):1333–1344, August 1999.

[21] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, Apr 1999.

[22] R. Wattenhofer, L. Li, P. Bahl, and Y. M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proc. IEEE Infocom*, 2001.

[23] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing & Communications*, 1999.