Routing and Traversal via Location Awareness in Ad-Hoc Networks

Evangelos Kranakis* Ladislav Stacho†

August 14, 2004

Abstract

We survey some recent results that make use of location awareness of the hosts of an ad-hoc network in order to provide for efficient information dissemination. We explore several new methodologies for constructing hop- and geometric-spanners in a distributed manner, discuss advantages and disadvantages of preprocessing the network topology, and outline several algorithms for efficient traversal and route discovery in ad-hoc networks.

1 Challenges in Ad-Hoc Networking

The current rapid growth in the spread of wireless devices of ever increasing miniaturization and computing power has greatly influenced the development of ad-hoc networking. Ad-hoc networks are wireless, self-organizing systems formed by co-operating nodes within communication range of each other that form temporary networks with a dynamic decentralized topology. It is desired to make a variety of services available (e.g., internet, GPS, service discovery) in such environments and our expectation is for a seamless and ubiquitous integration of the new wireless devices with the existing wired communication infrastructure. At the same time we anticipate the development of new wireless services that will provide solutions to a variety of communication

*School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada.

†Department of Mathematics, Simon Fraser University, 8888 University Drive, Burnaby, British Columbia, Canada, V5A 1S6.

needs (e.g., sensing and reporting an event, integrating hosts in a temporary network) among hosts without the interference of an existing infrastructure.

To comprehend the magnitude of our task and the challenges that need to be addressed it is important to understand that currently hosts of an ad-hoc network are energy limited and must operate in a bandwidth limited communication environment. Providing for efficient information dissemination in such an ad-hoc network is a significant challenge. The present paper is a survey with a threefold goal. First, to understand and propose a reasonable abstract model of communication that integrates well with the expected "location awareness" of the hosts, second to explain how by preprocessing the underlying wireless network we can simplify the communication infrastructure, and third propose solutions for achieving efficient information dissemination.

A brief outline of the paper is as follows. After discussing various abstract models in Section 2 we address how neighborhood proximity can be used to produce geometric and hop spanners in Section 3. Finally, in Section 4 we outline several algorithms for route discovery and traversal in undirected and directed ad-hoc networks.

2 Modelling Ad-hoc Networks

An ad-hoc network can be modelled as a set of points, representing network nodes (hosts), in 2-dimensional or 3-dimensional Euclidean space $\mathbb{R}^2$ or $\mathbb{R}^3$, with a relation among the nodes representing communication links (or wireless radio channels). One node is linked with another if the signal of the former can reach the latter. Hosts have limited computational and communication power. The computational power corresponds to the level of coding the host can accomplish as well as the amount of local memory accessible to the host. The communication power reflects how far a signal can reach and is determined by propagation characteristics of the radio channel in the environment as well as the power control capabilities of the host.

Modelling a radio channel is a nontrivial task due to possible signal loss or signal degeneracy caused by interference with other channels and blockages by physical obstructions. Radio propagation and interference models based on the physical layer have been developed in [12, 13]. These models provide accurate information on the capacity and limitations of an ad-hoc network. However they are too general for the design and analysis routing protocols

in higher layers. Thus, a simpler model that abstracts the wireless ad-hoc nature of the network is being used. In this model, the wireless network is modelled as a graph $G = (V, E)$ where V is the set of all hosts/nodes and E contains an edge joining u to v if the node u can directly transmit to v. Adjacency between nodes can be determined, for example, by a pass loss equation and a signal to noise formula [19]. If we assume that the transmission power of all the nodes is the same and fixed, then the graph G will be an undirected graph, called a *unit disk graph*. This is the standard theoretical model of wireless ad-hoc network. On the other hand, if hosts can adjust their transmission power and the transmission power is not the same for all hosts, then the resulting graph G will be a directed graph. The directed graph also models the signal loss in the wireless radio channels and as such can be viewed as a more realistic model of a wireless ad-hoc network.

2.1 Communication and information dissemination

An essential requirement of information dissemination in communication networks is that the nodes be endowed with a consistent addressing scheme. IP-addresses, so typical of traditional Internet networking, can be either setup at initialization or else acquired by means of an address configuration protocol. Unfortunately none of these techniques are suitable for wireless networking where addresses have to be built "on the fly" and are usually based on the geographic coordinates of the hosts. The latter can be acquired either directly from a participating satellite of an available GPS or via radio-location by exchanging signals with neighbouring nodes in the wireless system. Thus, the coordinates of the hosts form the foundation of any addressing scheme in such infrastructure-less networks. The corresponding model is then a unit disk graph together with an embedding (not necessarily planar) into $\mathbb{R}^2$. We refer to the graph with an embedding as *geometric graph* if we want to emphasize the fact that nodes are aware of their coordinates.

In general, ad-hoc networks are infrastructure-less, have no central components and the nodes may have unpredictable mobility patterns. In order to produce dynamic solutions that take into account the changing topology of the backbone the hosts should base their decision only on factors whose optimality depends on geographically local conditions.

A common approach to ad-hoc wireless network design is to separate the infrastructure design and the routing scheme design. However these two parts are closely interlaced as the choice of a particular infrastructure

control algorithm may have a considerable impact on the choice of the routing scheme. In Section 3, we discuss the infrastructure control techniques, and in Section 4 we survey some well-known routing techniques in wireless ad-hoc networks. Since hosts of a wireless ad-hoc network are embedded in 2-dimensional (or 3-dimensional) Euclidean space, many of these techniques borrow tools from combinatorial geometry.

3 Ad-Hoc Communication Infrastructure

Due to the host's limited resources in a wireless ad-hoc network, an important task is to compute and maintain a suitable topology over which high level routing protocols are implemented. One possible approach is to retain only a linear number of radio channels using some local topology control algorithm. A fundamental requirement on the resulting sparse network topology is that it has similar communication capabilities and power consumption as the original network. In the unit disk graph model, this directly leads to the problem of finding a sparse subgraph so that for each pair of nodes there exists an efficient path in the subgraph. A path is *efficient* if its Euclidean length or number of edges/hops is no more than some small constant factor (*length-stretch factor* in the former case and *hop-stretch factor* in the latter case) of the minimum needed to connect the nodes in the unit disk graph.

If our objective is to optimize the power consumption and the ad-hoc network has hosts with variable transmission ranges, then the resulting communication infrastructure should have good (small) length-stretch factor. Constructions of such topologies are based on various types of proximity graphs, and we review some of them in Subsection 3.1.

If the hosts of the wireless ad-hoc network have fixed transmission power, then the power consumption is minimized when the routes in the communication infrastructure minimize hops. We present a technique for maintaining communication infrastructure with optimal (up to a small constant) hop-stretch factor in Subsection 3.3.

Fundamental criteria for evaluation of the communication infrastructure include the following.

- *Bounded Degree* – each node has only a bounded number of neighbours in the subgraph. This is crucial since nodes have limited memory and power resources.

- *Localized Construction* – this is one of the most important requirements on the infrastructure control algorithm since nodes have only access to the information stored in nodes within a constant number of hops away. Thus, distributed local algorithms for maintenance of the topology are desirable.
- *Planarity* – this strong geometric requirement forces edges not to cross each other in the embedding. Many routing algorithms rely on planarity of the communication topology.
- *Sparseness* – the topology should be a sparse subgraph, i.e. the number of links should be linear in the number of nodes of the network. Sparseness enables most routing algorithms to run more efficiently. Note that planarity as previously mentioned implies sparseness.
- *Spanner* – nodes in the substructure should have distance "comparable" to their distance in the original network. In particular, a subgraph $S \subseteq G$ is a spanner of G if there is a positive constant t so that for every pair of nodes u and v, $d_S(u, v) \leq t \cdot d_G(u, v)$. This general definition gives rise to various types of spanners: if the distance function d measures the Euclidean length, then the subgraph is *geometric spanner* and the constant t is the *length-stretch factor*. On the other hand, if d is the number of edges/hops, then the subgraph is *hop spanner* and the constant t is the *hop-stretch factor*.

3.1 Geometric Spanners

In this section, we survey several results on constructing sparse communication topology (spanner) for ad-hoc networks with hosts of variable transmission ranges. Consequently the obtained spanners will be geometric spanners.

Suppose that P is a set of n points in the plane. Every pair (p, q) of points is associated with a planar region $S_{p,q}$ that specifies "the neighborhood association" of the two points. Let $\mathcal{S}$ be the set of neighborhood associations in the pointset. Formally, the neighborhood graph $G_{\mathcal{S},\mathcal{P}}$ of the planar pointset P is determined by a property $\mathcal{P}$ on the neighborhood associations:

1. $(p, q) \rightarrow S_{p,q} \subseteq \mathbb{R}^2$, for $p, q \in P$.
2. $\mathcal{P}$ is a property on $\mathcal{S} := \{S_{p,q} : p, q \in P\}$.

The graph $G_{\mathcal{S},\mathcal{P}} = (P, E)$ has the pointset P as its set of nodes and the set E of edges is defined by $(p, q) \in E \Leftrightarrow S_{p,q}$ has property $\mathcal{P}$. There are several

variants on this model including Nearest Neighbor Graph, Relative Neighbor Graph, and Gabriel Graph which we define in the sequel.

3.1.1 Nearest Neighbor Graph (NNG)

The edges of a nearest neighbor graph are determined by minimum distance. More precisely, for two points p and q, $(p, q) \in E \Leftrightarrow p$ is nearest neighbor of q. If $S_{p,q}$ is the disk with diameter determined by the points p and q then

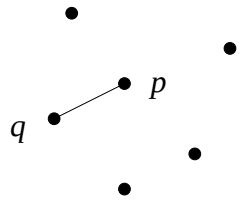

Figure 1: q is the nearest neighbor of p.

it is easy to see that the NNG is the same as the graph $G_{\mathcal{S},\mathcal{P}}$, where the property $\mathcal{P}$ is defined by $S_{p,q} = \emptyset$. Although the NNG incorporates a useful notion it has the disadvantage that the resulting graph may be disconnected. A generalization of the NNG is the k-Nearest Neighbor Graph (k-NNG), for some $k \geq 1$. In this case, $(p, q) \in E \Leftrightarrow p$ is k-th nearest neighbor of q or q is k-th nearest neigbhor of p.

3.1.2 Relative Neighbor Graph (RNG)

The neighborhood association in the Relative Neighbor Graph (RNG) is determined by the lune. Formally, the *lune* $L_{p,q}$ of p and q is the intersection of the open disks with radius the distance $d(p, q)$ between p and q and centred at p and q, respectively. The set of edges of the graph is defined by $(p, q) \in E \Leftrightarrow$ the lune $L_{p,q}$ does not contain any point in the pointset P.

3.1.3 Gabriel Graph (GG)

Gabriel graphs (also known as Least squares adjacency graphs) where introduced by Gabriel and Sokal in [11] and have been used for geographic analysis and pattern recognition. The region of association between two points p, q is specified by the disk with diameter determined by p and q. Formally, the set

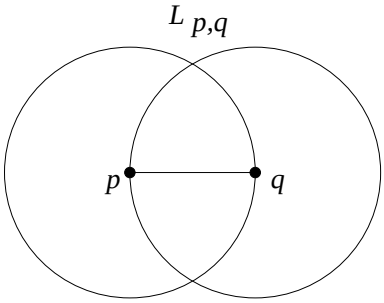

Figure 2: The lune defined by points p and q.

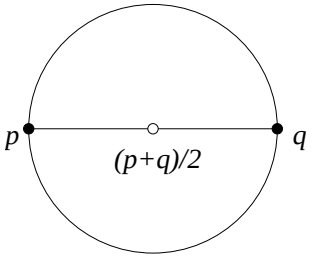

Figure 3: The circle with diameter the line segment determined by p, q and centred at the point $\frac{p+q}{2}$.

of edges of the Gabriel Graph is defined by $(p, q) \in E \Leftrightarrow$ the disk centred at $\frac{p+q}{2}$ and radius $\frac{d(p.q)}{2}$ does not contain any point in the pointset P.

3.1.4 Relation between the neighborhood graphs

We mention without proof the following theorem.

Theorem 1 (O'Rourke and Toussaint [20, 16]). *Let P be a planar pointset. Then the following inclusions are satisfied for the previously mentioned neighborhood graphs*

$$NNG \subset MST \subset RNG \subset GG \subset DT,$$

where MST denotes the Minimum spanning tree, and DT the Delaunay triangulation of the pointset P.

3.2 Tests for Preprocessing the Ad-Hoc Network

In this section, we describe two procedures for constructing a geometric spanner in a given ad-hoc network: the Gabriel test is due to Bose et al. [5] and the Morelia test is due to Boone et al. [3].

3.2.1 Gabriel Test

One of the most important tests for eliminating crossings in a wireless network is called Gabriel test which is applied to every link of the network. Assume that all nodes have the same transmission range R. Let A, B be two nodes whose distance is less than the transmission range R of the network. In the Gabriel test, if there is no node in the circle with diameter AB then

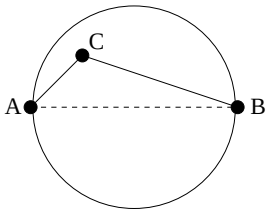

Figure 4: Eliminating an unnecessary link (dashed line AB) with the Gabriel test.

the link between A and B is kept. If however there is a node C in the circle with diameter AB, as depicted in Figure 4, then nodes A and B remove their direct link. Formally, the Gabriel test is as follows.

Gabriel Test.
Input: Pointset
Output: Gabriel Graph of Pointset

1: **for** for each node v **do**
2: **for** for each u neighbor of v **do**
3: **if** circle with diameter uv contains no other points in P **then**
4: remove the edge uv
5: **end if**
6: **end for**
7: **end for**

In particular, when A (respectively, B) is queried on routing data to B (respectively, A) the routing table at A (respectively, B) forwards the data through C (or some other similar node if more than one node is in the circle with diameter AB. The advantage of doing this rerouting of data is that the resulting graph is a planar spanner on which we can apply the face routing algorithm (to be described later) for discovering a route from source to destination.

Theorem 2 (Bose et al. [5]). *If the original network is connected then the Gabriel test produces a connected planar network.*

3.2.2 Morelia Test

The Gabriel test suffers from the multiple hop effect. Consider a set of pairwise mutually reachable nodes as depicted in the left-hand side of Figure 5. When we apply the Gabriel test the configuration in the right-hand side of

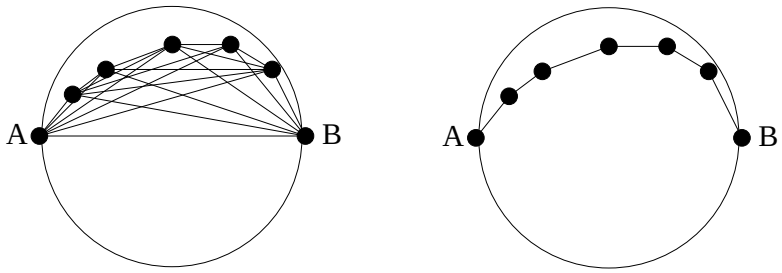

Figure 5: Multiple hop effect when eliminating a link (line segment AB) via the Gabriel test.

Figure 5 results. We can see that although nodes A and B could have reached each other directly in a single hop instead they must direct their data through a sequence of hops.

The Morelia test takes into account the "distance one" neighborhood of the nodes prior to deciding on whether it should delete an edge. It is similar to the Gabriel test in that given two nodes A and B it eliminates links based on the inspection of the circle with diameter AB. Unlike the Gabriel test it does not necessarily eliminate the direct link AB when it finds another node inside the circle with diameter AB. Instead, it verifies whether the nodes inside the circle create any crossing of the line AB. If no crossing is created

the line AB is kept, otherwise it is removed. The verification of the existence of crossing is done in most cases by inspecting only the neighborhood of nodes A and B at the transmission distance R. In a few cases, the neighborhood of some of the nodes in the circle around AB is inspected. We subdivide the area of the circle with diameter AB into four regions X_1, X_2, Y_1 and Y_2 (see Figure 6) as determined by an arc of radius R.

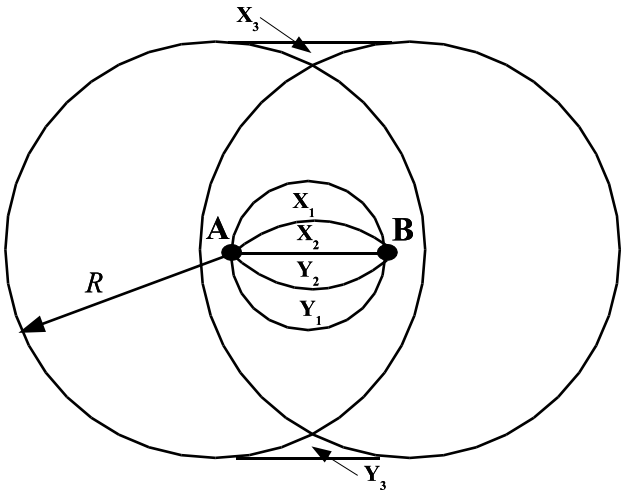

Figure 6: Morelia Test

Figure 7 depicts several scenarios being considered prior to determining whether or not an edge should be deleted. Details of the precise specification of the Morelia test applied to a link AB can be found in Boone et al. [3].

We can prove the following theorem.

Theorem 3 (Boone et al. [3]). *If the original network is connected then application of the Morelia test produces a connected planar spanner of the original network which contains the Gabriel graph of the network as a subgraph.*

3.3 Hop Spanners

In all constructions described in previous subsections, the computed topology is a sparse geometric spanner. If the hosts of the wireless network can adjust

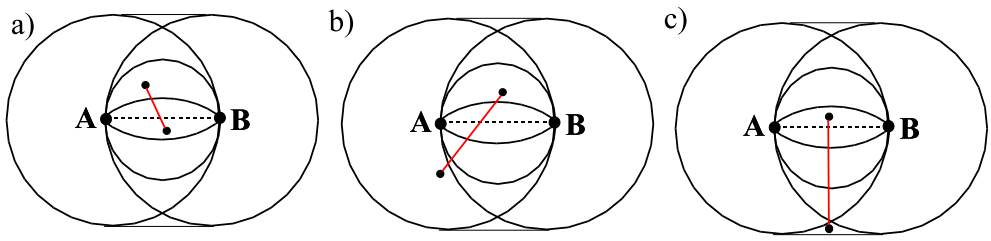

Figure 7: Examples of the Morelia test scenarios: a) AB deleted by rule 1, b) AB deleted by rule 2, c) AB deleted by rule 4.

their transmission power, then the topology will guarantee optimal power consumption. However the communication delay is not bounded.

To bound the communication delay, one must construct a topology with a small hop-stretch factor. Hop spanners were introduced by Peleg and Ullman in [18] and were used as network synchronizers. A survey of results on (sparse) hop spanners is given in [17]. The problem of finding a sparsest hop spanner with a given hop-stretch factor t was shown to be NP-complete for most values of t in general graphs, see [6]. In the case $t = 2$ (i.e., the hop length of a shortest path in a spanner is at most twice the hop length of a shortest path in the original graph) an approximation algorithm to construct a hop spanner with hop-stretch factor 2 is given in [14]. However, such a spanner cannot have a linear number of edges in general networks [14]. Recently a distributed non-constant hop-stretch factor $O(\log n)$ spanner algorithm appeared in [10].

For unit disk graphs, various hierarchical structures, based on the concepts of dominating sets and coverings, have been proposed. Although hop spanners in unit disk graphs can be made sparse, they cannot have every node of constant degree. This follows from a more general result in [2] where the following generalization of a hop spanner has been introduced.

Let S be a subgraph of G. We say that u and v are *quasi-connected* (in S) if there exists a path $(u, u_1, \ldots, u_k, v)$ in G with all but the last edge in $E(S)$. Such a path will be referred to as a *quasi-path* in S. Given two quasi-connected nodes u and v, their *quasi-distance* $qd_S(u, v)$ is the number of edges of a shortest $u - v$ quasi-path in S. A subgraph $S \subseteq G$ is a hop quasi-spanner of G if there is a positive constant t so that for every pair of

nodes u and v, $qd_S(u,v) \leq t(d_G(u,v)-1)+1$. Notice that according to this definition every hop spanner is also a hop quasi-spanner with the same hop-stretch factor. Thus, the notion of quasi-spanner generalizes the notion of hop spanner. Furthermore, in a wireless ad-hoc network it completely suffices to route a message to a host that is in the transmission range of the destination host. Indeed, if a host receives a message that is destined for it, it can accept the message, disregarding the fact that the sender may not be in its routing table based on the communication topology.

Now the following result shows that hop spanners must contain some nodes of large degree.

Theorem 4 (Berenbrink et al. [2]). *Let u be a node of a unit disk graph G. Let d and D be the number of nodes at distance 1 and 2 from u in G, respectively. Let H be a hop quasi-spanner of G with hop-stretch factor two. Then some node in G has degree at least $\sqrt{\frac{D}{d}}$.*

In the remainder of this section, we survey some known results on computing hop spanners with small hop-stretch factor. The most common approach is to first compute a dominating set of the wireless network G, then to add some connector nodes to transform the dominating set into a connected subgraph CDS of G, and finally to connect all remaining nodes to the nodes of CDS. The best result using this technique is the following.

Theorem 5 (Alzoubi et al. [1]). *There exists a local distributed algorithm that will compute a hop spanner of a unit disk graph with the hop-stretch factor 5. Moreover, the hop spanner is planar.*

From the practical point of view, the hop-stretch factor of the spanner from Theorem 5 is large. Another approach was taken in [2] where the hop quasi-spanner was proposed as the communication topology for wireless ad-hoc networks. The quasi-spanner is build on the concept of coverings: Every node selects a constant number of nodes covering all other nodes in the circle with radius 2 centered at the node. Since the algorithm is simple, we present it in detail.

The *k-neighbourhood* of a node v, denoted as $\mathrm{N}_G^k(v)$, is the set of all nodes at distance k from v. Similarly, we denote the set of all nodes at distance at most k from v as $\mathrm{N}_G^{\leqslant k}(v)$. A subset C of a set S is called a *covering* of S if for every point $s \in S$ there is a point $c \in C$ such that the distance between s and c is at most 1. Given a unit disk graph G and a node $u \in V$, a covering

of $\mathrm{N}_G^2(u)$ will be denoted as $T(u)$. A covering $T(u)$ is *minimal* if no other covering of $\mathrm{N}_G^2(u)$ is a subset of $T(u)$.

The following is a simple greedy procedure constructing a minimal covering.

Min-Covering(u, G)**.**
Output: minimal covering $T(u)$ of $\mathrm{N}_G^2(u)$

```
1: N := N_G^2(u)
2: T(u) = ∅
3: while N ≠ ∅ do
4:    choose v ∈ N
5:    T(u) := T(u) ∪ {v}
6:    for every x ∈ N_G^1(v) ∩ N do
7:       N := N − {x}
8:    end for
9: end while
```

The algorithm for constructing a hop quasi-spanner follows. The algorithm is very simple — every node will build a minimal covering around it.

Hop Quasi-Spanner.
Input: Unit disk graph $G = (V, E(G))$
Output: Spanning subgraph $K = (V, E(K))$ of G

```
1: for each node u do
2:    E(u) := ∅
3:    T(u) := Min-Covering(u, G)
4:    for every v ∈ T(u) do
5:       find a w ∈ N_G^1(u) ∩ N_G^1(v)
6:       E(u) := E(u) ∪ {(u, w), (w, v)}
7:    end for
8: end for
9: E(H) := ∪_{u∈V} E(u)
```

The hop quasi-spanner algorithm produces a sparse hop quasi-spanner. This is stated in the following theorem.

Theorem 6 (Berenbrink et al. [2]). *The graph K produced by the hop quasi-spanner algorithm is a hop quasi-spanner with the hop-stretch factor* 2 *and has at most* $36|V|$ *edges, i.e., the average degree of K is at most* 72*.*

4 Information Dissemination

After producing an underlying communication topology, an important task is how to disseminate information efficiently. For example, in routing we want to guarantee real-time message delivery and at the same time (1) avoid flooding the network, (2) discover an optimal path (the existence of such a path strongly depends on our choice of communication topology), (3) use only geographically local information. It is inevitable that we may not be able to satisfy all three goals at the same time. Traditional methods discover routes using a greedy approach in which the "next-hop" is determined by iteratively (hop-to-hop) reducing the value of a given function of a distance and/or angle determined by the source and destination hosts. Unfortunately this does not always work either because of loops (return to the source host without finding the destination) or voids (hosts with no neighbors closer to the destination). In this section, we look at the problem of determining in which networks it is possible to attain geographically local routing with guaranteed delivery.

4.1 Routing in Undirected Planar Graphs

In this section, we will consider planar geometric graphs as the communication topology and will describe several routing algorithms for them. Let G be a planar geometric graph. Recall that V denotes the set of nodes, and E the set of edges in G. We denote the set of faces of G by F. We will assume that nodes of G are in general position, i.e., no three are colinear.

4.1.1 Compass Routing

The first routing algorithm we consider is greedy in the sense that it always selects its next edge/hop on the basis of the smallest angle between the "current edge" and the "straight line" formed by the current node and the target node. More precisely the compass routing is achieved by the following algorithm.

Compass Routing Algorithm.
Input: connected planar graph $G = (V, E)$
Source node: s
Destination node: t

1: Start at source node $c := s$.

2: **repeat**
3: Choose an edge incident to current node c forming the smallest angle with straight line $c - t$ connecting c to t.
4: Traverse the chosen edge.
5: Go back to 3
6: **until** target t is found

In a deterministic setting, compass routing guarantees delivery if the planar graph is, for example, a Delaunay Triangulation (see Kranakis et al. [15]). It is easy to show that in a random setting, whereby the points of the set P are uniformly distributed over a given convex region, compass routing can reach the destination with high probability. In general, however, compass routing cannot guarantee delivery. This is not only due to possible voids (i.e., hosts with no neighbors closer to the destination) along the way but also inherent loops in the network (see Figure 8).

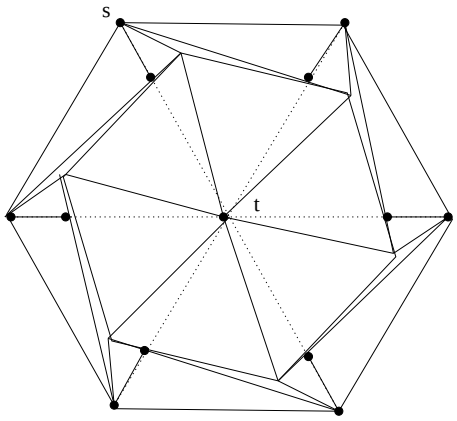

Figure 8: An example of a planar graph over which compass routing fails to find the destination.

In the next subsection, we describe a routing algorithm with guaranteed delivery on geometric planar graphs.

4.1.2 Face Routing Algorithm

Failure of compass routing to guarantee delivery is due to its "greedy" nature in order to minimize memory storage. Face routing overcomes this problem

by remembering the straight line connecting the source s to the target node t.

Face Routing Algorithm.
Input: Connected Geometric Planar Network $G = (V, E)$
Source node: s
Destination node: t

1: Start at source node $c := s$.
2: **repeat**
3: **repeat**
4: Determine face f incident to current node c and intersected by the straight line $s - t$ connecting s to t.
5: Select any of the two edges of f incident to c and traverse the edges of f in the chosen direction.
6: **until** we find the second edge, say xy, of f intersected by $s - t$.
7: Update the face f to the new face of the graph G that is "on the other side" of the edge xy.
8: Update current node c to either of the nodes x or y.
9: **until** target t is found.

An example of the application of face routing is depicted in Figure 9. Face routing always advances to a new face. We never traverse the same face

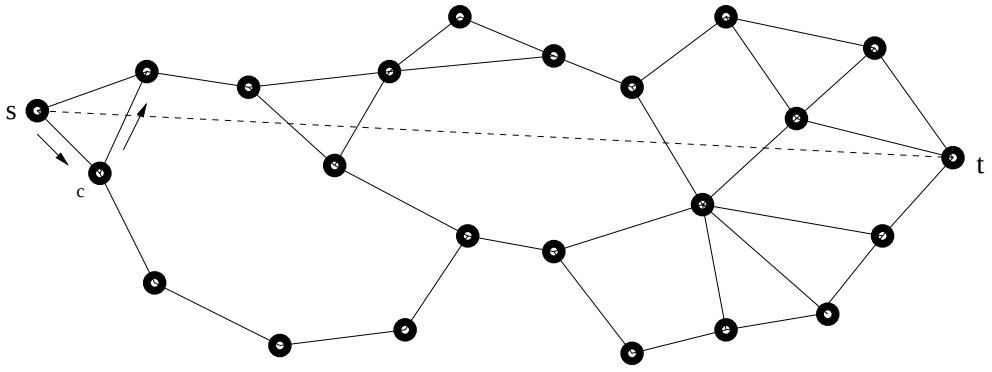

Figure 9: Face routing on a planar graph.

twice. Since each link is traversed a constant number of times we can prove the following theorem.

Theorem 7 (Kranakis et al. [15]). *Face routing in a geometric planar graph always guarantees delivery and traverses at most $O(n)$ edges.*

The reader may wish to consult [5] for additional variants of the face routing algorithm.

4.2 Routing in Directed Planar Graphs

As discussed earlier the unit disk graph is the graph of choice used in modelling ad-hoc networks when the communication links are bidirectional. However, bidirectionality cannot be assured in a system consisting of diverse hosts having variable power ranges, and nodes facing obstructions that may attenuate the signals. This gives rise to the "directed link model" which provides a natural direction between links as follows. If a node A can reach a node B (but it is not necessarily the case that B can also reach A) then we say that there is a directed link from A to B in the graph. This gives rise to an orientation in the network.

An *orientation* $\vec{}$ of a graph G is an assignment of a direction to every edge e of G. For an edge e with endpoints u and v, we write $e = (u, v)$ if its direction is from u to v. A graph G together with its orientation is denoted by $\vec{G}$. In this subsection, we will consider two classes of directed planar geometric graphs: Eulerian and Outerplanar. For both classes, we present a simple routing algorithm.

4.2.1 Eulerian Networks

Consider a connected planar geometric graph $G = (V, E)$. If G is oriented, then the face routing algorithm will not necessarily work since some edges may be directed in an opposite direction while traversing a face. We describe a simple method from [7] for routing a message to the other end of an oppositely directed edge in Eulerian geometric networks. The idea is simple: imitate the face routing algorithm but when an edge that needs to be traversed is misdirected find a way to circumvent it using only "constant" memory and geographically local information.

We say that $\vec{G}$ is *Eulerian* if for every node $u \in V$, the number of edges outgoing from u equals the number of edges ingoing into u, i.e., the size of $N^+(u) = \{x, (u, x) \in E\}$ equals the size of $N^-(u) = \{y, (y, u) \in E\}$. Now suppose that $\vec{G}$ is an Eulerian planar geometric graph. For a given node u of

$\vec{G}$, we order edges (u, x) where $x \in N^+(u)$ clockwise around u starting with the edge closest to the vertical line passing through u. Similarly we order edges (y, u) where $y \in N^-(u)$ clockwise around u (see Figure 10).

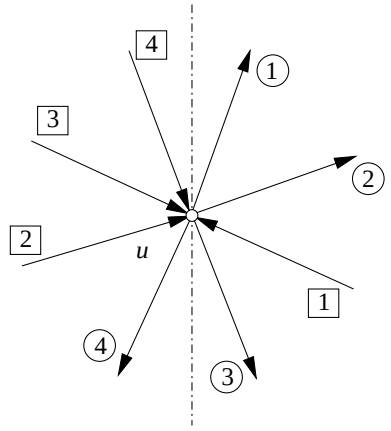

Figure 10: Circled numbers represent the ordering on outgoing edges, squared numbers on ingoing ones.

Let $e = (y, u)$ be the i-th ingoing edge to u in $\vec{G}$. The function $\mathbf{succ(e)}$ will return a pointer to the edge (u, x) so that (u, x) is the i-th outgoing edge from u. For an illustration of the function see Figure 11. Again, this function is easy to implement using only local information. Obviously, the function $\mathbf{succ}()$ is injective, and thus, for every edge $e = (u, v)$ of $\vec{G}$, we can define a closed walk by starting from $e = (u, v)$ and then repeatedly applying the function $\mathbf{succ}()$ until we arrive at the same edge $e = (u, v)$. Since $\vec{G}$ is Eulerian, the walk is well defined and finite. We call such a walk a *quasi-face* of $\vec{G}$.

We modify the face routing algorithm from [15] so that it will work on Eulerian planar geometric graphs. This requires extending the face traversal routine so that whenever the face traversal routine wants to traverse an edge $e = (u, v)$ that is oppositely directed, we traverse the following edges in this order: $\mathbf{succ(e)}, \mathbf{succ(e)^2}, \ldots, \mathbf{succ(e)^k}$, so that $\mathbf{succ(e)^{k+1}} = \mathbf{(u, v)}$. After traversing $\mathbf{succ(e)^k}$, the routine resumes to the original traversal of the face. Formally the algorithm is as follows.

Eulerian Directed Graph Algorithm.
Input: Connected Eulerian geometric graph $\vec{G} = (V, E)$

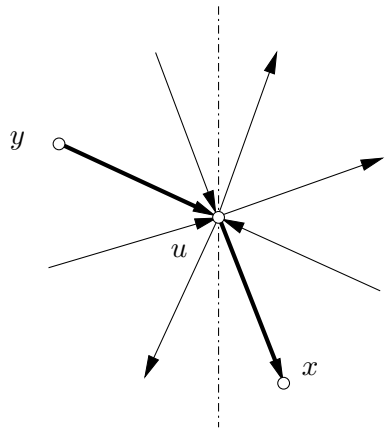

Figure 11: In this example the ingoing edge (y, u) is third, so the chosen outgoing edge (u, x) is also third. Both these edges are depicted bold.

Source node: s
Destination node: t

1: $v \leftarrow s$ {Current node = source node.}
2: **repeat**
3: Let f be a face of G with v on its boundary that intersects the line v-t at a point (not necessarily a node) closest to t.
4: **for all** edges xy of f **do**
5: **if** $xy \cap v\text{-}t = p$ and $|pt| < |vt|$ **then**
6: $v \leftarrow p$
7: **end if**
8: **end for**
9: Traverse f until reaching the edge xy containing the point p.
10: **until** $v = t$

We can prove the following theorem.

Theorem 8 (Chavez et al. [8]). *The Eulerian Directed Graph Algorithm will reach t from s in at most $O(n^2)$ steps.*

4.2.2 Outerplanar Networks

If $\vec{G}$ is not Eulerian, then the routing algorithm above will fail. It is an important open problem to design routing algorithms for general directed

geometric graphs. A fundamental question is whether such an algorithm exists for directed planar geometric graphs. In all cases, the directed graphs must be strongly connected so that we have a directed path guaranteed.

In this subsection, we describe a routing algorithm from [8] that works on outerplanar graphs. A planar geometric graph G is *outerplanar* if a single face (called outer-face) contains all the nodes. We will assume that this face is a convex polygon in $\mathbb{R}^2$. For a given triple of nodes x, y, and z, let $V_{\curvearrowleft}(x, y, z)$ (respectively, $V_{\curvearrowright}(x, y, z)$) denote the ordered set of nodes distinct from x and z that are encountered while moving from y counterclockwise (respectively, clockwise) around the outer-face of G until either x or z is reached; see Figure 12.

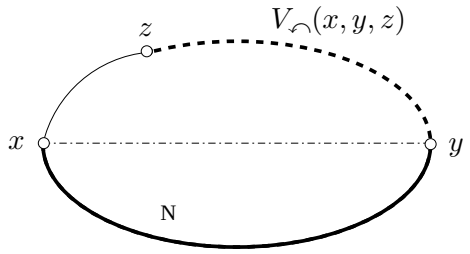

Figure 12: The dashed part of the outer face represents the nodes in $V_{\curvearrowleft}(x, y, z)$ and the bold solid part represents nodes in $V_{\curvearrowright}(x, y, z)$, respectively. Note that y belongs to both these sets and is in fact the first element of those sets.

Now consider an orientation $\vec{G}$ of the geometric graph G and let

$$N_{\curvearrowleft}(x, y, z) = V_{\curvearrowleft}(x, y, z) \cap N^+(x) \text{ and } N_{\curvearrowright}(x, y, z) = V_{\curvearrowright}(x, y, z) \cap N^+(x).$$

If $N_{\curvearrowleft}(x, y, z) \neq \emptyset$, let $v_{\curvearrowleft}(x, y, z)$ denote the first node in $N_{\curvearrowleft}(x, y, z)$. Similarly we define $v_{\curvearrowright}(x, y, z)$ as the first node in $N_{\curvearrowright}(x, y, z)$, if it exists. A geometric network with fixed orientation is *strongly connected* if for every ordered pair of its nodes, there is a (directed) path joining them.

The intuitive idea of the algorithm is to start at the source node s. It specifies the two nodes v_1, v_2 adjacent to the current node c such that the straight line determined by nodes c and t lies within the angle v_1cv_2. It traverses one of the nodes and backtracks if necessary in order to update its current position. Formally, the algorithm is as follows.

Outer Planar Directed Graph Algorithm.
Input: Strongly connected outerplanar geometric graph $\vec{G} = (V, E)$
Source node: s
Destination node: t

1: $v \leftarrow s$ {Current node = source node.}
2: $v_{\curvearrowleft}, v_{\curvearrowright} \leftarrow s$ {counterclockwise and clockwise bound = starting node.}
3: **while** $v \neq t$ **do**
4: **if** $(v, t) \in E$ **then**
5: $v, v_{\curvearrowleft}, v_{\curvearrowright} \leftarrow t$ {Move to t.}
6: **else if** $N_{\curvearrowleft}(v, t, v_{\curvearrowleft}) \neq \emptyset$ and $N_{\curvearrowright}(v, t, v_{\curvearrowright}) = \emptyset$ **then** {No-choice node; greedily move to the only possible counterclockwise direction toward t.}
7: $v, v_{\curvearrowleft} \leftarrow v_{\curvearrowleft}(v, t, v_{\curvearrowleft})$
8: **else if** $N_{\curvearrowleft}(v, t, v_{\curvearrowleft}) = \emptyset$ and $N_{\curvearrowright}(v, t, v_{\curvearrowright}) \neq \emptyset$ **then** {No-choice node; greedily move to the only possible clockwise direction toward t.}
9: $v, v_{\curvearrowright} \leftarrow v_{\curvearrowright}(v, t, v_{\curvearrowright})$
10: **else if** $N_{\curvearrowleft}(v, t, v_{\curvearrowleft}) \neq \emptyset$ and $N_{\curvearrowright}(v, t, v_{\curvearrowright}) \neq \emptyset$ **then** {Decision node; first take the "counterclockwise" branch but remember the node for the backtrack purpose.}
11: $b \leftarrow v$; $v, v_{\curvearrowleft} \leftarrow v_{\curvearrowleft}(v, t, v_{\curvearrowleft})$
12: **else if** $N_{\curvearrowleft}(v, t, v_{\curvearrowleft}) = \emptyset$ and $N_{\curvearrowright}(v, t, v_{\curvearrowright}) = \emptyset$ **then** {Dead-end node; backtrack to the last node where a decision has been made. No updates to $v_{\curvearrowleft}$ and $v_{\curvearrowright}$ are necessary.}
13: **if** $v \in V_{\curvearrowleft}(t, b, t)$ **then**
14: **while** $v \neq b$ **do**
15: $v \leftarrow v_{\curvearrowleft}(v, b, v)$
16: **end while**
17: **end if**
18: **if** $v \in V_{\curvearrowright}(t, b, t)$ **then**
19: **while** $v \neq b$ **do**
20: $v \leftarrow v_{\curvearrowright}(v, b, v)$
21: **end while**
22: **end if**
23: $v, v_{\curvearrowright} \leftarrow v_{\curvearrowright}(v, t, v_{\curvearrowright})$ {Take the "clockwise" branch toward t.}
24: **end if**
25: **end while**

We have the following theorem.

Theorem 9 (Chavez et al. [8]). *The Outer Planar Directed Graph Algo-*

rithm will reach t from s in at most $2n - 1$ *steps.*

4.3 Traversal of Non-Planar Graphs

Network traversal is a technique widely used in networking for processing the nodes, edges, etc, of a network in some order. For example it may involve reporting each node, edge, and face of a planar graph exactly once, in order to apply some operation to each. As such it can be used to discover network resources, implement security policies, and report network conditions. Although traversal can be used to discover routes between two hosts, in general it will be less efficient than routing since it cannot guarantee that its "discovery process" will be restricted to employing only information relevant to routing.

DFS (Depth First Search) of the primal nodes and edges or dual faces and edges of the graph is the usual approach followed for implementing traversal but it cannot be implemented without using mark bits on the nodes, edges, or faces, and a stack or queue. In this subsection we discuss a traversal technique from [7] that is applicable to a class of non-planar networks (to be defined below) and which is an improvement of a technique introduced by de Berg et al. [9] and further elaborated by Bose and Morin [4].

A *quasi-planar graph* $G = (V, E)$ has nodes embedded in the plane and partitioned into $V_p \cup V_c = V$ so that

- nodes in V_p induce a connected planar graph P,
- the outer-face of P does not contain any node from V_c or edge of $G - P$, and
- no edge of P is crossed by any other edge of G.

An example of a quasi planar graph is depicted in Figure 13.

We will refer to the graph P as an *underlying planar subgraph* and to its faces as *underlying faces.* The notion of nodes and edges is explicit in the definition of quasi-planar graph, however, the notion of faces is not. To define the notion of a face, we need to introduce some basic functions on quasi-planar graphs.

A node u is uniquely determined by the pair $[x, y]$, of its horizontal and vertical coordinates. Every edge $e = uv$ is stored as two oppositely directed edges (u, v) and (v, u). The functions $\mathbf{xcor}(\mathbf{v})$ (respectively, $\mathbf{ycor}(\mathbf{v})$) will return the horizontal (respectively, vertical) coordinate of the node v and $\mathbf{rev}(\mathbf{e})$ will return a pointer to the edge (v, u). Similarly the function $\mathbf{succ}(\mathbf{e})$

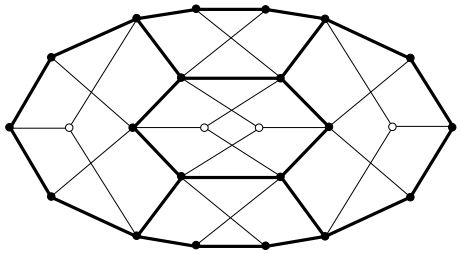

Figure 13: An example of a quasi-planar graph that satisfies the Left-Neighbor Rule. The filled nodes are in V_p and bold edges are edges of the underlying planar subgraph P.

will return a pointer to the edge (v, x) so that (v, x) is the first edge counter-clockwise around v starting from the edge (v, u), and the function **pred**(**e**) will return a pointer to the edge (y, u) so that (u, y) is the first edge clockwise around u starting from the edge (u, v). For an illustration of these functions see Figure 14.

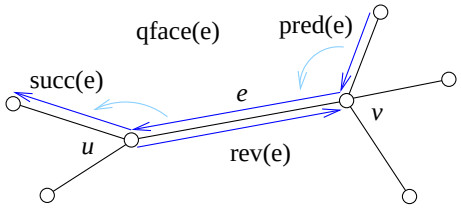

Figure 14: Illustration of basic functions on quasi-planar graphs.

For every edge $e = (u, v)$ of G, we can define a closed walk by starting from $e = (u, v)$ and then repeatedly applying the function **succ**() until we arrive at the same edge $e = (u, v)$. Such a walk is called a *quasi-face* of G and the set of all quasi-faces of G is denoted by F. The function **qface**(**e**) will return a pointer to the quasi-face determined by edge $e = (u, v)$.

A quasi-planar graph G satisfies the *Left-Neighbor Rule* if every node $v \in V_c$ has a neighbor u so that **xcor**(**u**) $<$ **xcor**(**v**). For an example of G that satisfies the Left-Neighbor Rule see Figure 13.

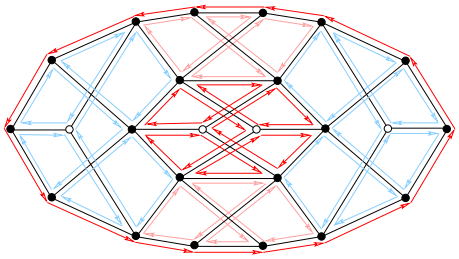

Figure 15: A quasi-planar graph and its six quasi-faces.

Quasi Planar Graph Traversal Algorithm

As in de Berg et al. [9], the general idea of the algorithm is to define a total order $\preceq$ on all edges in E. This gives rise to a unique predecessor for every quasi-face in F. The predecessor relationship imposes a virtual directed tree $G(F)$. The algorithm (due to Chavez et al. [7]) will search for the root of $G(F)$ and then will report quasi-faces of G in DFS order on the tree $G(F)$. For this, a well-known tree-traversal technique is used in order to traverse $G(F)$ using $O(1)$ additional memory.

In order to define the virtual tree $G(F)$, we determine a unique edge, called an *entry edge*, in each quasi-face. We first define a total order on all edges in E. We write $u \ll v$ if $(\mathbf{xcor}(\mathbf{u}), \mathbf{ycor}(\mathbf{u})) \leq (\mathbf{xcor}(\mathbf{v}), \mathbf{ycor}(\mathbf{v}))$ by lexicographic comparison of the numeric values using $\leq$. For an edge $e = (u, v)$, let

$$\mathbf{left}(\mathbf{e}) = \begin{cases} u, & \text{if } u \ll v \\ v, & \text{otherwise} \end{cases}, \quad \mathbf{right}(\mathbf{e}) = \begin{cases} v, & \text{if } u \ll v \\ u, & \text{otherwise} \end{cases},$$

and $\check{u} = [\mathbf{xcor}(\mathbf{u}), \mathbf{ycor}(\mathbf{u}) - \mathbf{1}]$. Now let $\mathbf{key}(\mathbf{e})$ be the 5-tuple

$$\begin{aligned} \mathbf{key}(\mathbf{e}) = &(\mathbf{xcor}(\mathbf{left}(\mathbf{e})),\ \mathbf{ycor}(\mathbf{left}(\mathbf{e})), \\ &\measuredangle\mathbf{left\check{}}(\mathbf{e})\mathbf{left}(\mathbf{e})\mathbf{right}(\mathbf{e}),\ \mathbf{xcor}(\mathbf{u}),\ \mathbf{ycor}(\mathbf{u})). \end{aligned}$$

By $\measuredangle abc$ we always refer to the counter-clockwise angle between rays ba and bc with b being the apex of the angle. It follows by our assumption that edges cannot cross vertices that if two edges $e \neq e'$ have the same first three values in their $\mathbf{key}()$, then $e' = e^-$ and hence their last two values in $\mathbf{key}()$ cannot both be the same. Hence it follows that $e = e'$ if and only if

$\mathbf{key}(\mathbf{e}) = \mathbf{key}(\mathbf{e}')$. We define the total order $\preceq$ by lexicographic comparison of the numeric $\mathbf{key}()$ values using $\leq$. For a quasi-face $f \in F$, we define

$$\mathbf{entry}(\mathbf{f}) = \mathbf{e} \in \mathbf{f} \ : \ \mathbf{e} \preceq \mathbf{e}' \text{ for all } \mathbf{e}' \neq \mathbf{e} \in \mathbf{f},$$

i.e., $\mathbf{entry}(\mathbf{f})$ is the minimum edge (with respect to the order $\preceq$) on the quasi-face f. Such an edge e will be called the *entry edge* of f. Note that this function is easy to implement using the function $\mathbf{succ}()$, and the total order $\preceq$ using only $O(1)$ memory.

The main algorithm for traversal from [7] is as follows.

Quasi-Planar Traversal Algorithm.
Input: $e = (u, v)$ of $G(V, E)$;
Output: List of nodes, edges, and quasi-faces of G.

repeat {find the minimum edge e_0}
  $e \leftarrow \mathbf{rev}(\mathbf{e})$
  while $e \neq \mathbf{entry}(\mathbf{qface}(\mathbf{e}))$ **do**
    $e \leftarrow \mathbf{succ}(\mathbf{e})$
  end while
until $e = e_0$
$p \leftarrow \mathbf{left}(\mathbf{e})$
repeat {start the traversal}
  $e \leftarrow \mathbf{succ}(\mathbf{e})$
  let $e = (u, v)$ and let $\mathbf{succ}(\mathbf{e}) = (\mathbf{v}, \mathbf{w})$
  if p is contained in $cone(u, v, w)$ **then** {report u if necessary}
    report u
  end if
  if $|up| < |vp|$ or ($|up| = |vp|$ and $\overrightarrow{up} < \overrightarrow{vp}$) **then** {report e if necessary}
    report e
  end if
  if $e = \mathbf{entry}(\mathbf{qface}(\mathbf{e}))$ **then** {report e and return to parent of $\mathbf{qface}(\mathbf{e})$}
    report qface$(\mathbf{e})$
    $e \leftarrow \mathbf{rev}(\mathbf{e})$
  else {descend to children of $\mathbf{qface}(\mathbf{e})$ if necessary}
    if $\mathbf{rev}(\mathbf{e}) = \mathbf{entry}(\mathbf{qface}(\mathbf{rev}(\mathbf{e})))$ **then**
      $e \leftarrow \mathbf{rev}(\mathbf{e})$
    end if
  end if

25: **until** $e = e_0$
26: **report qface**($\mathbf{e_0}$)

We have the following theorem.

Theorem 10 (Chavez et al. [7]). *The Quasi-Planar Traversal Algorithm reports each node, (undirected) edge, and quasi-face of a quasi-planar graph G that satisfies the Left-Neighbor Rule exactly once in $O(|E| \log |E|)$ time.*

5 Conclusion

In this paper we provided a survey of recent results on information dissemination via location awareness in ad-hoc networks. Although routing and traversal may be wrongfully considered as overworked research topics we believe that information dissemination in ad-hoc networks provides new challenges that go beyond trivial extensions of existing or old results. We expect that exploring tradeoffs between location awareness and reducing preprocessing time for simplifying a wireless system is of vital importance to our understanding of the efficiency of information dissemination. Future studies in a variety of graph models (oriented and otherwise) should provide better clues to maintaining a seamless, ubiquitous and well-integrated communication infrastructure.

Acknowledgments

Many thanks to the Morelia group (Edgar Chavez, Stefan Dobrev, Jarda Opartny, and Jorge Urrutia) for many hours of enjoyable discussions on the topics reviewed in this paper and to Danny Krizanc for his comments on a draft of the paper. Research of both authors is supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants.

References

[1] K. Alzoubi, X.Y. Li, Y. Wang, P.J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14:1–14, 2003.

[2] P. Berenbrink, T. Friedetzky, J. Manuch, and L. Stacho. (Quasi) spanner in mobile ad hoc networks. submitted.

[3] P. Boone, E. Chavez L., Gleitzky, E. Kranakis, J. Opartny, G. Salazar, and J. Urrutia. Morelia test: Improving the efficiency of the gabriel test and face routing in ad-hoc networks. In *Proceedings of SIROCCO 2004, Springer Verlag, LNCS, to appear*, 2004.

[4] P. Bose and P. Morin. An improved algorithm for subdivision traversal without extra storage. *International Journal of Computational Geometry and Applications*, 12(4):297–308, 2002.

[5] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of Discrete Algorithms and Methods for Mobility (DIALM'99), pages 48–55*, 1999.

[6] L. Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics*, 48:187–194, 1994.

[7] E. Chavez, S. Dobrev, E. Kranakis, J. Opartny, L. Stacho, and J. Urrutia. Route discovery with constant memory in oriented planar geometric networks. In *In Proceedings of Algosensors 2004, Springer Verlag, LNCS, to appear*, 2004.

[8] E. Chavez, S. Dobrev, E. Kranakis, J. Opartny, L. Stacho, and J. Urrutia. Traversal of a quasi-planar subdivision without using mark bits. In *In WMAN (workshop on Wireless Mobile Adhoc Networks), IPDPS, Santa Fe, New Mexico, April 26-30*, 2004.

[9] M. de Berg, M van Kreveld, R. van Oostrum, and M. Overmars. Simple traversal of a subdivision without extra storage. *International Journal of Geographic Information Systems*, 11:359–373, 1997.

[10] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating

sets and linear-size skeletons. In *Proc. of 14nt ACM-SIAM Symposium on Discrete Algorithms (SODA 03)*, 2003.

[11] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systemic Zoology*, 18:259–278, 1969.

[12] M. Grossglauser and D. Tse. Mobility increases the capacity of wireless adhoc networks. In *Proc. of IEEE Infocom*, 2001.

[13] P. Gupta and P. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Technology*, 46:388–404, 2000.

[14] G. Kortsarz and D. Peleg. Generating sparse 2-spanners. *J. of Algorithms*, 17:222–236, 1994.

[15] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of 11th Canadian Conference on Computational Geometry*, pages 51–54, August 1999.

[16] J. O'Rourke and G. T. Toussaint. Patterm recognition. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 797–813. CRC Press, New York, 1997.

[17] D. Peleg and A.A. Schaffer. Graph spanners. *J. Graph Theory*, 13:99–116, 1989.

[18] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Computing*, 18:740–747, 1989.

[19] T. Rappaport. *Wireless communications: Principles and practices*. Prentice Hall, 1996.

[20] G. T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–266, 1980.