Routing with Guaranteed Delivery in *ad hoc* Wireless Networks*

Prosenjit Bose
School of Computer Science
Carleton University
jit@scs.carleton.ca

Pat Morin
School of Computer Science
Carleton University
morin@scs.carleton.ca

Ivan Stojmenović
Computer Science, SITE
University of Ottawa
ivan@site.uottawa.ca

Jorge Urrutia
Computer Science, SITE
University of Ottawa
jorge@site.uottawa.ca

Abstract

We consider routing problems in *ad hoc* wireless networks modeled as *unit graphs* in which nodes are points in the plane and two nodes can communicate if the distance between them is less than some fixed unit. We describe the first distributed algorithms for routing that do not require duplication of packets or memory at the nodes and yet guaranty that a packet is delivered to its destination. These algorithms can be extended to yield algorithms for broadcasting and geocasting that do not require packet duplication. A byproduct of our results is a simple distributed protocol for extracting a planar subgraph of a unit graph. We also present simulation results on the performance of our algorithms.

1 Introduction

Mobile *ad hoc* networks (MANETs) consist of wireless hosts that communicate with each other in the absence of fixed infrastructure. Two nodes in a MANET can communicate if the distance between them is less than the minimum of their two broadcast ranges [1]. For health and efficiency reasons, it is generally not possible (or desirable) for all hosts in a MANET to be able to communicate with each other directly. Thus, sending messages between two hosts in a MANET may require routing the message through intermediate hosts.

In many cases, MANETs are pieced together in an uncontrolled manner, changes in topology are frequent and unstructured, and hosts may not know the topology of the entire network. In this paper, we consider routing in MANETs for which hosts know nothing about the network except their location and the locations of the hosts to which they can communicate directly. GPS (global positioning system) provides each host with its geographic location and global timing [11]. In particular, we consider the case in which all hosts have the same broadcast range.

Let S be a set of points in the plane. Then the *unit graph* $U(S)$ is a geometric graph that contains a vertex for each element of S. An edge (u, v) is present in $U(S)$ if and only if $dist(u, v) \leq 1$, where $dist(x, y)$ denotes the Euclidean distance between x and y. Unit graphs are a reasonable mathematical abstraction of wireless networks in which all nodes have equal broadcast ranges.

In this paper we describe algorithms for routing on unit graphs which do not require global information about $U(S)$. Each vertex $v \in U(S)$ represents a transmission station, and has no information about $U(S)$ except the set of nodes $N(v)$ to which it is adjacent. A packet that is stored at vertex v can be transmitted to any vertex in $N(v)$. In accordance with other papers [1, 3, 7, 9], it is assumed that the source knows from the beginning the exact geographical position of the destination.

In the *routing* problem, the source v_{src} and destination v_{dst} are points of S and v_{dst} must receive a message originating at v_{src}. In the *geocasting* problem [6, 11] the source v_{src} is a point in S while the destination r_{dst} is a region, and all vertices in r_{dst} must receive a message originating at v_{src}. In this work we take r_{dst} to be a disk, but our algorithms generalize to arbitrary convex regions. Broadcasting is then a special case of geocasting in which r_{dst} is a disk with infinite radius.

Previous algorithms for online routing in unit graphs can be broadly classified into two categories:

Greedy algorithms apply some type of greedy path-finding heuristic that does not guarantee that a packet ultimately reaches (all of) its destination(s). These include the *geographic distance routing* (GEDIR) algorithm of Lin and Stojmenović [9], the *directional routing* (DIR), a.k.a, *compass routing* algorithm of Basagni *et al* [1],

*This work was partly funded by the Natural Sciences and Engineering Research Council of Canada.

Ko and Vaidya [6], and Kranakis *et al* [8], the MFR algorithm of Takagi and Kleinrock [15], and their 2-hop variants [9].

Flooding algorithms use some type of controlled packet duplication mechanism to ensure that every destination receives at least one copy of the original packet . These are exemplified by the *location-aided routing* (LAR) protocols of Ko and Vaidya [7, 6]. In order for flooding algorithms to terminate, packets in the network must remember which packets they have previously seen.

In contrast, our routing algorithms always guarantee that a packet will be delivered to (all of) its intended recipient(s) so long as the unit graph $U(S)$ is static and connected. Our algorithms do not make use of any memory at the nodes of $U(S)$ and require only that a packet carry a small constant amount of information in addition to its message. Our algorithms also never require duplication of a packet, so that at any point in time there is exactly one copy of each message in the network. Although the delivery is guaranteed only for fixed graphs, it may be possible to apply our algorithms to moving hosts, in conjunction with location update techniques [1, 7].

Our algorithms work by finding a connected planar subgraph of $U(S)$ and then applying routing algorithms for planar graphs on this subgraph. In Section 2 we show how to find a connected planar subgraph of $U(S)$ in an online distributed manner. In Section 3 we describe algorithms for routing, broadcasting, and geocasting in planar graphs. In Section 4 we describe simulation results for our algorithm. Finally, in Section 5 we summarize and conclude with open problems in the area.

2 Extracting a Connected Planar Subgraph

In this section we describe a distributed algorithm for extracting a connected planar subgraph from $U(S)$. In order to run the algorithm, the only information needed at each node is the position of each of its neighbors in $U(S)$. Our algorithm works by computing the intersection of $U(S)$ with a well-known planar graph.

Let $disk(u, v)$ be the disk with diameter (u, v). Then, the *Gabriel graph* [5] $GG(S)$ is a geometric graph in which the edge (u, v) is present if and only if $disk(u, v)$ contains no other points of S. The following lemma shows that the Gabriel graph is useful for extracting a connected subgraph from $U(S)$.

Lemma 1. *If $U(S)$ is connected then $GG(S) \cap U(S)$ is connected.*

Proof. It is well known that a minimum spanning tree $MST(S)$ is a subset of $GG(S)$ [13]. Thus, we need only prove that $MST(S) \subseteq U(S)$ if $U(S)$ is connected. Assume for the sake of contradiction that $MST(S)$ contains an edge (u, v) whose length is greater than 1. Removing this edge from $MST(S)$ produces a graph with two connected components, $C_u(S)$ and $C_v(S)$. Since $U(S)$ is connected it contains an edge (w, x) of length not greater than 1 such that $w \in C_u(S)$ and $x \in C_v(S)$. By replacing the edge (u, v) with (w, x) in $MST(S)$ we obtain a connected graph on S with weight less than $MST(S)$, a contradiction. □

Let (u, v) be an edge of $U(S)$ such that $(u, v) \notin GG(S)$. Then, by the definition of $GG(S)$ there exists a point w that is contained in the disk with u and v as diameter, and this point acts as a *witness* that $(u, v) \notin GG(S)$. The following lemma shows that every such edge can be identified and eliminated by u and v using only local information.

Lemma 2. *Let u and v be points of $U(S)$ such that $(u, v) \notin GG(S)$ and let w be a witness to this. Then $(u, w) \in U(S)$ and $(v, w) \in U(S)$.*

Proof. Let m be the midpoint of (u, v). Then $dist(u, m) \leq 1/2$, $dist(v, m) \leq 1/2$ and $dist(w, m) \leq 1/2$. Therefore, by the triangle inequality, $dist(u, w) \leq 1$, $dist(v, w) \leq 1$ and (u, w) and (v, w) are in $U(S)$. □

Thus, upon reaching a vertex $v \in S$, a packet can eliminate the edges incident on v that are not in $U(S) \cap GG(S)$ by simply eliminating any edge that is not in $GG(N(v) \cup \{v\})$. This leads to the following algorithm that is executed by each vertex $v \in S$.

Algorithm: GABRIEL

```
1: for each $u \in N(v)$ do
2:   if $disk(u, v) \cap (N(v) \setminus \{u, v\}) \neq \emptyset$ then
3:     delete $(u, v)$
4:   end if
5: end for
```

Lemma 1 guarantees that if we apply this algorithm to each vertex of S then the resulting graph is connected. Since $GG(S)$ is planar [12, 10, 5], the resulting graph is also planar. As described above, the algorithm requires $O(d^2)$ time, where d is the degree of v. By using efficient algorithms for constructing the Voronoi diagram (VD) and Delaunay triangulation (DT) [12, 13] of $N(v) \cup \{v\}$, and keeping edges of DT that intersect corresponding edges of VD [10, 5], this can be reduced to $O(d \log d)$.

Theorem 1. *If $U(S)$ is connected then algorithm* GABRIEL *computes a connected planar subgraph of $U(S)$. The cost of the computation performed at vertex $v \in S$ is $O(d \log d)$ where d is the degree of v.*

Remark: More realistically, the elimination of edges not in $GG(S)$ could be done when the network is initialized or when changes in network topology occur.

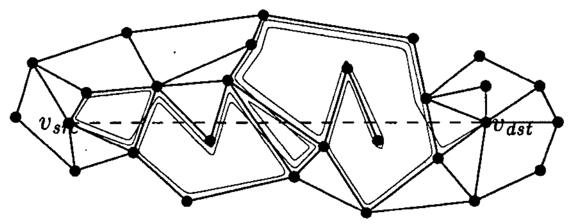

Figure 1: Routing from v_{src} to v_{dst} using FACE-1.

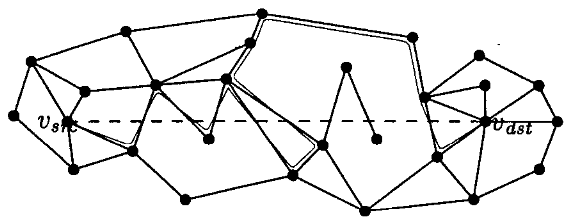

Figure 2: Routing from v_{src} to v_{dst} using FACE-2.

3 Routing in Planar Graphs

In this section we describe online algorithms for routing, broadcasting, and geocasting in a connected planar graph G. Since we have shown that a connected planar subgraph of $U(S)$ is easily computable by a routing algorithm, these algorithms also apply to unit graphs.

3.1 Routing

In this section we describe algorithms for routing in planar graphs. The first algorithm, called FACE-1, is due to Kranakis *et al* [8]. The second algorithm, called FACE-2, is a modification of their algorithm that performs better in practice.

A connected planar graph G partitions the plane into *faces* that are bounded by polygonals made up of edges of G. Given a vertex v on a face F, the boundary of F can be traversed in the counterclockwise (clockwise if F is the outer face) direction using the well-known *right hand rule* [2]. Treating this face traversal technique as a subroutine, Kranakis *et al* [8] give the following algorithm for routing a packet from v_{src} to v_{dst}.

Algorithm: FACE-1

```
1: $p \leftarrow v_{src}$
2: repeat
3:    let $F$ be the face of $G$ with $p$ on its boundary that
      intersects line segment $(p, v_{dst})$
4:    for each edge $(u, v)$ of $F$ do
5:       if (u,v) intersects $(p, v_{dst})$ in a point $p'$ and
         $dist(p', v_{dst}) < dist(p, v_{dst})$ then
6:          $p \leftarrow p'$
7:       end if
8:    end for
9:    Traverse $F$ until reaching the edge $(u, v)$ contain-
      ing $p$
10: until $p = v_{dst}$
```

The operation of algorithm FACE-1 algorithm is illustrated in Figure 1. The following theorem summarizes the performance of this algorithm.

Theorem 2 (KSU 1999 [8]). *Algorithm* FACE-1 *reaches v_{dst} after at most $4|E|$ steps, where $|E|$ is the number of edges in G.*

Notice that this algorithm traverses the entire face F to determine the point p', and then must return to the point p'. The bound $4|E|$ stated in the theorem can be reduced to $3|E|$ by having the return trip to p' be along the shorter of the two possible paths around F. However, in practice, as we will show in Section 4, the following modified version of FACE-1 works even better.

Algorithm: FACE-2

```
1: $p \leftarrow v_{src}$
2: repeat
3:    let $F$ be the face of $G$ with $p$ on its boundary that
      intersects $(p, v_{dst})$
4:    traverse $F$ until reaching an edge $(u, v)$ that in-
      tersects $(p, v_{dst})$ at some point $p' \neq p$
5:    $p \leftarrow p'$
6: until $p = v_{dst}$
```

The operation of FACE-2 is illustrated in Figure 2. Clearly this algorithm also terminates in a finite number of steps, since the distance to v_{dst} is decreasing during each round. However, in pathological cases it may visit $\Omega(n^2)$ edges of G.

Theorem 3. *Algorithm* FACE-2 *reaches v_{dst} in a finite number of steps.*

3.2 Broadcasting

De Berg *et al* [4] describe an algorithm for enumerating all the faces, edges, and vertices of a connected embedded planar graph G. It requires no memory at the nodes of the graph and uses only $O(1)$ additional memory in the packet that is traveling around the network. The algorithm works by defining a spanning tree on the faces of G and performing depth first search on this spanning tree in $O(n^2)$ time, where n is the number of vertices of G.[1] This algorithm can be made into a routing algorithm that allows a single packet to visit every vertex in G. We refer to this algorithm as BROADCAST.

Theorem 4. *In at most $O(n^2)$ steps algorithm* BROADCAST *terminates after having visited every vertex of G.*

[1]Actually, the algorithm has running time $O\left(\sum_i |F_i|^2\right)$ where $|F_i|$ denotes number of edges in the ith face of G.

3.3 Geocasting

De Berg *et al* also extend their results to window queries in which all the faces intersecting a rectangular or circular query region r_{dst} are to be visited. To start their algorithm, a vertex of a face that intersects r_{dst} must be given as part of the input.

By applying Algorithm FACE-1, such a face can be found in $O(n)$ steps by setting the value of v_{dst} to the center of the query region. The algorithm should terminate when it reaches a vertex v contained in r_{dst} or when it can no longer make progress, i.e., it visits the same face twice. In the first case we apply the algorithm of de Berg *et al* to have the packet visit every vertex in the query region, while in the second case we can quit, since there is no vertex of G contained in the query region. We call this algorithm GEOCAST.

Theorem 5. *In at most $O(n+k^2)$ steps algorithm* GEOCAST *terminates after having visited every vertex of G contained in r_{dst}, where k is the complexity of all faces of G that intersect r_{dst}.*

Remark: The delivery time for a message in the broadcasting and geocasting algorithms can be improved in practice by traversing subtrees of the spanning tree in parallel, at the cost of having several copies of the same packet in the network simultaneously.

4 Experimental Results

In this section we measure the quality of the paths found by our routing algorithms. Our test sets consist of randomly constructed unit graphs. Test cases were generated by uniformly selecting n points in the unit square as vertices, sorting all the $n(n-1)/2$ interpoint distances and setting the value of a "unit" to achieve the desired average degree. Any such random graph that did not result in a connected graph was rejected. For each graph generated, routing was performed between all $n(n-1)$ ordered pairs of vertices in the graph. Every data point shown in our graphs is the average of 200 independent trials conducted on 200 different randomly generated graphs. The results of these trials are given as 95% confidence intervals in Appendix A.

For comparison purposes the performance of our algorithms were measured against, and in combination with, geographic distance routing (GEDIR) as described by Lin and Stojmenović [9]. The GEDIR algorithm is a greedy algorithm that always moves the packet to the neighbour of the current vertex whose distance to the destination is minimized. The algorithm fails when the packet crosses the same edge twice in succession. The GEDIR algorithm was chosen for comparison purposes because, of the three basic algorithms tested by Lin and Stojmenović, GEDIR had the best performance in terms of delivery rate and average dilation (defined below).

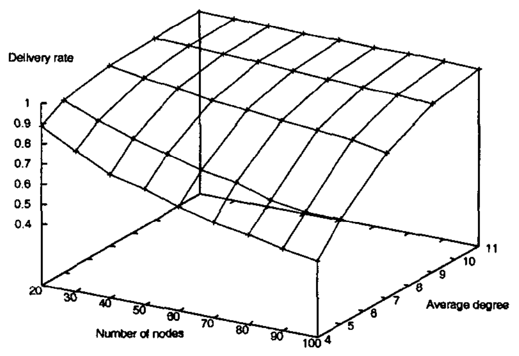

Figure 3: Delivery rates for the GEDIR algorithm.

The experiments measured two quantities. Let X be the set of pairs of vertices $(u,v) \in G$, $u \neq v$ such that routing algorithm $\mathcal{A}$ succeeds in finding a path from u to v and let $|X|$ denote the cardinality of X. The *delivery rate* of $\mathcal{A}$ is defined as

$$DR_{\mathcal{A}}(G) = |X|/(n(n-1)) \ .$$

Note that, because our algorithms guarantee the delivery of a packet, they have a delivery rate of 1. The *average dilation* of $\mathcal{A}$ is defined as

$$AD_{\mathcal{A}}(G) = (1/|X|) \sum_{(u,v)\in X} AP(u,v)/SP(u,v) \ ,$$

where $AP(u,v)$ is the number of edges in the path from u to v found by $\mathcal{A}$ and $SP(u,v)$ is the number of edges in the shortest path from u to v. Note that having a low average dilation is only useful if the delivery rate is high since an average dilation of 1 is easily achieved by (for example) an algorithm that only succeeds in routing between two nodes if they are directly adjacent.

To illustrate the importance of having guaranteed delivery of messages, Figure 3 shows the delivery rate of GEDIR on graphs with varying average degrees and number of nodes. These results show that delivery failures are not uncommon with the GEDIR algorithm, and in very sparse graphs delivery rates can be as low as 50%. I.e., there are some vertices from which half of the graph is unreachable.

Figure 4 compares the FACE-1 algorithm with the FACE-2 algorithm in terms of average dilation for varying average degrees and number of nodes. Not surprisingly, FACE-2 outperforms FACE-1 due to the fact that it does not require the packet to travel all the way around each face. What may be surprising is that the average dilation for both strategies seems to increase as the average degree increases. This can be explained by the fact that the subgraph $GG(S) \cap U(S)$ on which these algorithms operate is a planar graph and therefore has

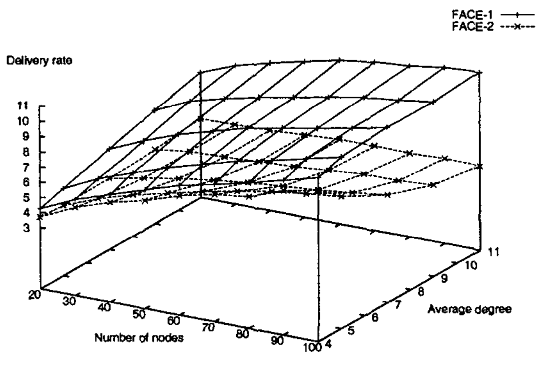

Figure 4: Average dilation of the FACE-1 and FACE-2 algorithms.

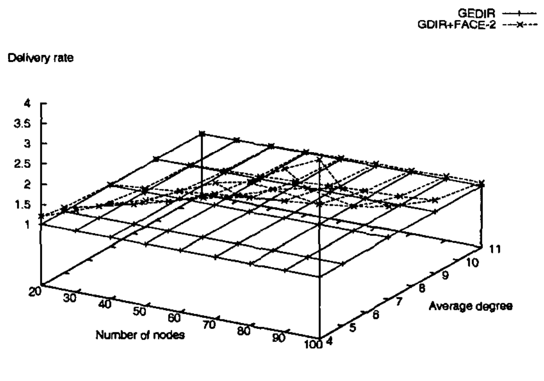

Figure 5: Average dilation of the GEDIR and GEDIR+FACE-2 algorithms.

average degree at most 6, but they are being compared to the shortest path in $U(S)$ whose average degree is increasing. Thus, the algorithms are handicapped "from the start."

Although these observations may lead one to believe that algorithms FACE-1 and FACE-2 are not very good on their own, they may nevertheless be useful in combination with another algorithm. We tested two such combinations and compared their average dilation with the average dilation of GEDIR.

Figure 5 shows the results of combining the GEDIR algorithm with FACE-2 by applying the GEDIR algorithm until it either failed or reached the destination. If the GEDIR algorithm failed, routing was then completed using the FACE-2 algorithm. In this scenario FACE-2 can be viewed as acting as a *backup* for the GEDIR algorithm. We refer to this algorithm as GEDIR+FACE-2.

Figure 6 shows the results of applying GEDIR until the packet reaches a node v such that all of v's neighbours are further from the destination than v is. The FACE-2 algorithm was then applied until the packet

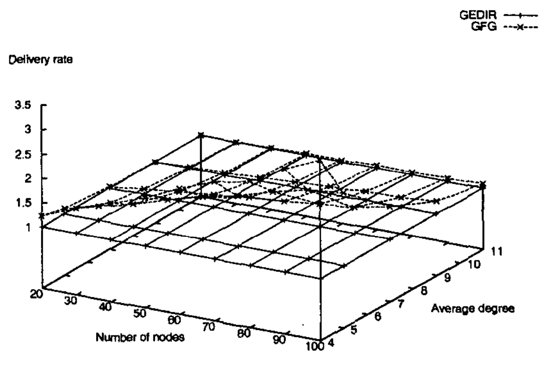

Figure 6: Average dilation of the GEDIR and GFG algorithms.

reached another vertex u that was strictly closer to the destination than v at which point the GEDIR algorithm was resumed. In this scenario, FACE-2 can be seen as a means of overcoming local minima in the objective function (distance to the destination). We refer to this algorithm as GFG.

Both these hybrid algorithms exhibit similar performance with the GFG algorithm showing a slight advantage in very sparse graphs. These results show that the average dilation of GEDIR is consistently low, but this comes at the price of low delivery rate in sparse graphs. On the other hand, the combined algorithms sometime have high average dilation, but this only occurs when the delivery rate of GEDIR is low and the combined algorithms are often forced to apply the FACE-2 algorithm. The combined algorithm simultaneously enjoys the advantages guaranteed delivery in sparse graphs and low average dilation in dense graphs.

5 Conclusions

We have described algorithms for routing, broadcasting, and geocasting in unit graphs. The algorithms do not require duplication of packets or memory at the nodes of the graph and yet guarantee that a packet is always delivered to (all of) its destination(s). The empirical results for our routing algorithms suggest that although the FACE-1 and FACE-2 algorithms are not very efficient on their own, they can be useful in conjunction with simpler algorithms that do not guarantee delivery.

The BROADCAST and GEOCAST algorithms are probably not very applicable in practice due to their quadratic message count and delivery time behaviour. An interesting open problem that is currently under investigation is whether or not algorithms exist that do not require memory at the nodes of $U(S)$ and takes subquadratic time to visit all vertices of $U(S)$.

Results for static networks like those in this and other papers [9] help in finding the most promising candidates for the design of routing protocols in mobile networks. There are a number of directions in which the work presented here can be extended and/or generalized, including results for dynamically changing networks, networks in three-dimensional space, nodes with unequal transmission ranges, and power-aware routing schemes [14].

References

[1] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, pages 76–84, 1998.

[2] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* Elsevier North-Holland, 1976.

[3] D. Camara and A. F. Loureiro. A novel routing algorithm for ad hoc networks. Manuscript, 1999.

[4] 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.

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

[6] Y.-B. Ko and N. H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. Technical Report TR-98-018, Texas A&M University, September 1998.

[7] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, pages 66–75, 1998.

[8] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*, 1999. To appear.

[9] X. Lin and I. Stojmenović. Geographic distance routing in ad hoc wireless networks. Technical Report TR-98-10, SITE, University of Ottawa, December 1998.

[10] D. W. Matula and R. R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis*, 12:205, July 1980.

[11] J. C. Navas and T. Imielinski. Geocast - geographic addressing and routing. In *ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'97)*, pages 66–76, 1997.

[12] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams.* John Wiley and Sons, 1992.

[13] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry.* Springer-Verlag, New York, 1985.

[14] I. Stojmenović. Power-aware routing in ad hoc wireless networks. Technical Report TR-98-11, SITE, University of Ottawa, December 1998.

[15] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

A Simulation Results

This appendix presents the results of simulations in tabular form. The variable d is the average degree of the graph and the variable n is the number of vertices in the graph.

d	4	5	7	9	11
n					
20	0.89 ± 0.0178	0.95 ± 0.0110	0.99 ± 0.0049	1.00 ± 0.0020	1.00 ± 0.0000
30	0.79 ± 0.0210	0.88 ± 0.0168	0.95 ± 0.0139	0.99 ± 0.0047	1.00 ± 0.0009
40	0.70 ± 0.0203	0.84 ± 0.0199	0.95 ± 0.0123	0.99 ± 0.0053	1.00 ± 0.0038
50	0.68 ± 0.0222	0.79 ± 0.0211	0.92 ± 0.0161	0.98 ± 0.0072	0.99 ± 0.0042
60	0.62 ± 0.0212	0.74 ± 0.0215	0.91 ± 0.0159	0.96 ± 0.0105	0.99 ± 0.0037
70	0.57 ± 0.0172	0.70 ± 0.0233	0.88 ± 0.0199	0.96 ± 0.0089	0.99 ± 0.0049
80	0.54 ± 0.0168	0.65 ± 0.0239	0.86 ± 0.0184	0.96 ± 0.0096	0.99 ± 0.0052
90	0.51 ± 0.0179	0.63 ± 0.0216	0.85 ± 0.0204	0.94 ± 0.0122	0.99 ± 0.0047
100	0.47 ± 0.0157	0.61 ± 0.0185	0.81 ± 0.0208	0.93 ± 0.0144	0.98 ± 0.0057

Table 1: 95% confidence intervals for delivery rates of GEDIR.

d	4	5	7	9	11
n					
20	1.01 ± 0.0013	1.01 ± 0.0010	1.00 ± 0.0006	1.00 ± 0.0003	1.00 ± 0.0000
30	1.01 ± 0.0011	1.01 ± 0.0013	1.01 ± 0.0010	1.00 ± 0.0006	1.00 ± 0.0002
40	1.01 ± 0.0013	1.01 ± 0.0013	1.01 ± 0.0011	1.00 ± 0.0007	1.00 ± 0.0006
50	1.01 ± 0.0013	1.02 ± 0.0013	1.01 ± 0.0009	1.01 ± 0.0008	1.00 ± 0.0006
60	1.02 ± 0.0012	1.02 ± 0.0013	1.02 ± 0.0013	1.01 ± 0.0010	1.01 ± 0.0006
70	1.02 ± 0.0015	1.02 ± 0.0012	1.01 ± 0.0009	1.01 ± 0.0009	1.01 ± 0.0007
80	1.02 ± 0.0011	1.02 ± 0.0015	1.02 ± 0.0011	1.01 ± 0.0010	1.01 ± 0.0008
90	1.02 ± 0.0012	1.02 ± 0.0012	1.02 ± 0.0012	1.01 ± 0.0009	1.01 ± 0.0009
100	1.02 ± 0.0013	1.02 ± 0.0011	1.02 ± 0.0011	1.02 ± 0.0010	1.01 ± 0.0007

Table 2: 95% confidence intervals for average dilation of GEDIR.

d	4	5	7	9	11
n					
20	4.27 ± 0.0911	4.74 ± 0.0838	5.63 ± 0.1025	6.42 ± 0.1040	7.15 ± 0.1171
30	5.26 ± 0.1094	5.88 ± 0.1116	6.60 ± 0.1229	7.49 ± 0.1312	8.10 ± 0.1291
40	6.02 ± 0.1254	6.70 ± 0.1388	7.47 ± 0.1448	8.02 ± 0.1524	8.62 ± 0.1514
50	6.83 ± 0.1150	7.40 ± 0.1493	8.11 ± 0.1661	8.44 ± 0.1613	9.25 ± 0.1581
60	7.56 ± 0.1238	7.99 ± 0.1351	8.75 ± 0.1893	9.07 ± 0.2025	9.69 ± 0.2179
70	8.09 ± 0.1511	8.69 ± 0.1647	9.08 ± 0.2184	9.44 ± 0.2121	9.97 ± 0.1947
80	8.62 ± 0.1426	9.15 ± 0.1843	9.68 ± 0.2420	9.71 ± 0.1828	10.18 ± 0.1762
90	9.24 ± 0.1484	9.79 ± 0.1419	10.12 ± 0.2562	10.17 ± 0.2364	10.42 ± 0.2047
100	9.78 ± 0.1605	10.28 ± 0.1852	10.57 ± 0.2596	10.54 ± 0.2766	10.62 ± 0.2012

Table 3: 95% confidence intervals for average dilation of FACE-1.

d	4	5	7	9	11
n					
20	3.69 ± 0.1691	3.62 ± 0.1863	3.71 ± 0.1881	3.90 ± 0.1762	4.11 ± 0.1989
30	4.70 ± 0.2117	4.64 ± 0.2065	4.24 ± 0.2273	4.28 ± 0.1954	4.29 ± 0.1855
40	5.48 ± 0.1947	5.17 ± 0.2473	4.59 ± 0.2213	4.26 ± 0.1845	4.19 ± 0.1856
50	6.11 ± 0.2216	5.63 ± 0.2554	4.93 ± 0.2341	4.28 ± 0.1836	4.43 ± 0.1686
60	6.79 ± 0.2564	6.09 ± 0.2560	5.22 ± 0.2642	4.59 ± 0.2334	4.50 ± 0.2275
70	7.45 ± 0.2891	6.69 ± 0.2891	5.29 ± 0.2868	4.67 ± 0.2286	4.52 ± 0.1981
80	7.74 ± 0.2585	7.13 ± 0.3522	5.66 ± 0.3077	4.70 ± 0.1818	4.49 ± 0.1707
90	8.58 ± 0.3291	7.47 ± 0.3143	5.92 ± 0.3304	4.91 ± 0.2264	4.50 ± 0.1775
100	9.02 ± 0.3510	7.64 ± 0.3272	6.18 ± 0.3495	5.12 ± 0.2713	4.53 ± 0.1766

Table 4: 95% confidence intervals for average dilation of FACE-2.

d	4	5	7	9	11
n					
20	1.21 ± 0.0373	1.10 ± 0.0280	1.03 ± 0.0131	1.01 ± 0.0042	1.00 ± 0.0000
30	1.51 ± 0.0579	1.32 ± 0.0496	1.13 ± 0.0381	1.02 ± 0.0179	1.00 ± 0.0036
40	1.84 ± 0.0682	1.48 ± 0.0665	1.17 ± 0.0391	1.05 ± 0.0169	1.02 ± 0.0119
50	2.08 ± 0.0970	1.69 ± 0.0779	1.29 ± 0.0581	1.07 ± 0.0228	1.04 ± 0.0158
60	2.45 ± 0.1172	1.92 ± 0.0911	1.36 ± 0.0687	1.14 ± 0.0423	1.04 ± 0.0139
70	2.86 ± 0.1262	2.23 ± 0.1111	1.46 ± 0.0817	1.16 ± 0.0360	1.06 ± 0.0209
80	3.08 ± 0.1136	2.53 ± 0.1443	1.56 ± 0.0878	1.17 ± 0.0350	1.06 ± 0.0218
90	3.50 ± 0.1661	2.69 ± 0.1378	1.66 ± 0.1051	1.25 ± 0.0547	1.07 ± 0.0233
100	3.92 ± 0.1736	2.87 ± 0.1392	1.85 ± 0.1282	1.33 ± 0.0763	1.09 ± 0.0250

Table 5: 95% confidence intervals for average dilation of GEDIR+FACE-2.

d	4	5	7	9	11
n					
20	1.22 ± 0.0368	1.12 ± 0.0259	1.03 ± 0.0130	1.01 ± 0.0053	1.00 ± 0.0001
30	1.53 ± 0.0574	1.32 ± 0.0463	1.14 ± 0.0323	1.03 ± 0.0136	1.01 ± 0.0036
40	1.77 ± 0.0655	1.46 ± 0.0576	1.18 ± 0.0369	1.06 ± 0.0160	1.02 ± 0.0132
50	1.99 ± 0.0932	1.66 ± 0.0777	1.27 ± 0.0483	1.08 ± 0.0198	1.05 ± 0.0189
60	2.30 ± 0.1139	1.85 ± 0.0867	1.36 ± 0.0602	1.14 ± 0.0384	1.04 ± 0.0121
70	2.61 ± 0.1218	2.05 ± 0.0915	1.41 ± 0.0684	1.16 ± 0.0311	1.06 ± 0.0182
80	2.75 ± 0.1061	2.26 ± 0.1189	1.50 ± 0.0750	1.16 ± 0.0280	1.06 ± 0.0199
90	3.12 ± 0.1504	2.43 ± 0.1298	1.57 ± 0.0905	1.23 ± 0.0455	1.08 ± 0.0228
100	3.48 ± 0.1748	2.51 ± 0.1219	1.74 ± 0.1134	1.30 ± 0.0673	1.09 ± 0.0198

Table 6: 95% confidence intervals for average dilation of GFG.