Fast Localized Delaunay Triangulation*

Filipe ARAÚJO
Universidade de Lisboa
filipius@di.fc.ul.pt†

Luís RODRIGUES
Universidade de Lisboa
ler@di.fc.ul.pt †

Abstract

A localized Delaunay triangulation owns the following interesting properties in a wireless *ad hoc* setting: it can be built with localized information, the communication cost imposed by control information is limited and it supports geographical routing algorithms that offer guaranteed convergence. This paper presents a localized algorithm that builds a graph called planar localized Delaunay triangulation, $PLDel$, known to be a good spanner of the unit disk graph, UDG. Unlike previous work, our algorithm builds $PLDel$ in a single communication step, maintaining a communication cost of $O(n \log n)$, which is within a constant of the optimum. This represents a significant practical improvement over previous algorithms with similar theoretical bounds. Furthermore, the small cost of our algorithm makes feasible to use $PLDel$ in real systems, instead of the Gabriel or the Relative Neighborhood graphs, which are not good spanners of UDG.

Keywords: Wireless *ad hoc* networks, Location-based routing schemes, Delaunay triangulation

1 Introduction

Wireless *ad hoc* networks are networks where nodes communicate with neighbors within some range using a wireless link. Nodes of a wireless network typically operate on batteries and thus have relatively few memory and energy resources. It is therefore utterly important to rely on routing schemes with small state and communication overhead. This requirement can be met by a *localized* routing scheme, where nodes only maintain information about other nodes within a limited neighborhood. On the other hand, for the sake of efficiency, a routing scheme should be *competitive*, i.e., any path found by the scheme should be at most c times longer than the shortest path. However, Kuhn *et al.* proved that no localized scheme can be c-competitive [9]. Still, a localized routing scheme can guarantee convergence, while achieving competitive path lengths in most cases.

*This work was partially supported by LaSIGE and by the FCT project INDIQoS POSI/CHS/41473/2001 via POSI and FEDER funds. Selected portions of this report will be published in the proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS), December 15-17 2004, Grenoble, France.

†Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa, Campo Grande, Edifício C6, 1749-016 Lisboa, Portugal.

One way of achieving competitive routing is to build a global Delaunay Triangulation [2]. Unfortunately, building such a graph is not a viable solution to the routing problem in *ad hoc* wireless networks, because: *i)* edges may be longer than communication range; *ii)* it cannot be built locally and therefore, communication cost is too high. Hence, our approach is to build a planar graph (i.e., without intersection of edges) as dense as possible ($O(n)$ edges), using Delaunay triangulations, but in a localized fashion. The point of having a dense graph is to use routing algorithms that achieve good hop count performance, while planarity is necessary to ensure convergence.

In literature, there are several algorithms that build Delaunay triangulations for routing purposes, e.g., [12, 13, 6, 10]. The algorithm in [13] builds a subgraph of the global Delaunay triangulation that only includes some of the edges within communication range of nodes; [12], [6] and [10] build a denser graph, with global communication cost of $O(n \log n)$, $O(n^2)$ and $O(n^2)$, respectively. While [6] and [10] are not optimal, [12] involves 4 communication steps to build the final subgraph, which may be prohibitive in practical systems. Hence, in this paper, we improve on the work of Li *et. al* [12], by presenting an algorithm that is considerably simpler and yet builds the same *Planar Localized Delaunay Triangulation* graph ($PLDel$), with the same asymptotic communication cost, but with just a single communication step (we define a communication step as the period required for sending and then receiving one or more messages which are not causally related).

Therefore, our algorithm is well suited to wireless environments for the following reasons: i) it is very efficient as it requires just one communication step; ii) it is applicable to dynamic and asynchronous settings (see Section 6); iii) it is localized, only requiring nodes to receive information broadcast by direct neighbors, thus requiring a communication cost within a small constant of the optimum (assuming that a beacon message of $O(\log n)$ bits in an n-node network is necessary per node); iv) it requires nodes to keep track of only a constant number of neighbors in the average; v) under the constraint of preserving planarity, it builds a graph with good density (see Section 5).

The rest of the paper is structured as follows. For self-containment we provide a short overview of necessary concepts in Section 2. In Section 3 we provide a survey on related work on wireless networks and Delaunay triangulations. In Section 4 we describe our algorithm and prove its correctness. In Section 5, we experimentally evaluate our algorithm. The appplication of the algorithm in dynamic settings is discussed in Section 6. Finally, Section 7 concludes the paper.

2 Preliminaries

We assume that nodes can determine their own position and the position of their neighbors. Given a set of nodes V in a two dimensional space, we model a wireless *ad hoc* network as a unit disk graph, $UDG(V)$, which is comprised of all nodes V and all edges connecting pairs of nodes of V whose distance is at most 1, i.e., in this model, two nodes A and B are direct neighbors (or simply *neighbors*) if and only if $||AB|| \leq 1$. Nodes A and B are k-hop neighbors if they can reach each other in k or fewer hops. Throughout this paper, we will use the following notation: a triangle defined by nodes A, B and C is represented as $\triangle ABC$; an angle ($< \pi$) between edges AB and AC defined at A is interchangeably represented as $\angle BAC$ or $\angle CAB$; the circle whose diameter is defined by two nodes A and B is represented as $d(A, B)$; the circumcircle defined by node A, B and C is represented as $\bigcirc ABC$.

The *Gabriel graph* (GG) is comprised of all edges AB such that $d(A, B)$ does not contain any other node of V. The edges of a GG are called *Gabriel edges*. The *Relative neighborhood graph* (RNG) is comprised of all edges AB such that there is no node C for which $||AC|| < ||AB||$ and $||BC|| < ||AB||$ (i.e., node C, cannot be simultaneously closer to A and B than A and B are from each other). It should be noted that RNG is a subgraph of GG. The *Delaunay triangulation* (DT) of a node set V, represented as $Del(V)$, is the set of edges satisfying the "empty circle" property: edge AB belongs to the triangulation if and only if there is a circle containing A and B, but not containing any other node. An important property of $Del(V)$ that will be of use to us, states that the circumcircle of a triangle does not contain any node of V. Under the UDG model, a complete Delaunay triangulation may not exist, because some edges may be longer than 1 and therefore, we refer to $UDel(V) = Del(V) \cap UDG(V)$ instead.

In this paper we will use the definition proposed in [12] of *k-localized Delaunay graph over a node set V*, $LDel^{(k)}(V)$. $LDel^{(k)}(V)$ is comprised of two types of edges (not longer than 1): *i)* all edges from the GG; and *ii)* edges of all triangles ABC for which there are no nodes inside $\bigcirc ABC$ reachable by A, B or C in k or fewer hops. Li *et al.* [12] proved that $LDel^{(k)}(V)$ is planar for $k \geq 2$, but edges may intersect for $k = 1$. $PLDel(V)$ [12, 10] is defined as a planar graph comprised of all triangles of $LDel^{(1)}(V)$, *except* intersecting triangles that do not belong to $LDel^{(2)}(V)$. Moreover, Li *et al.* [12] proved that $UDel(V)$ is a $(4\sqrt{3}\pi)/9$-spanner of $UDG(V)$ and that $LDel^{(k)}(V) \supseteq UDel(V)$. Hence $PLDel(V)$ and $LDel^{(k)}(V)$, for all k, are also $(4\sqrt{3}\pi)/9$-spanners of $UDG(V)$.

3 Related Work

In literature, we can find several algorithms that build Delaunay triangulations, e.g. [11, 4, 16]. Of particular interest to us are the algorithms that allow Delaunay triangulations to be computed in an incremental way [1, 17], as new nodes that arrive later do not force a recomputation of the entire triangulation.

In [14], Liebeherr *et al.* proposed an algorithm to build a complete non-localized Delaunay triangulation that serves as an overlay network on top of IP. However, direct application of this algorithm to the more complex setting of a wireless environment is not possible since Delaunay neighbors may not be able to communicate if their distance is greater than 1. In the context of wireless networks, geographic routing algorithms like greedy and compass have received wide attention in literature [8, 18]: these algorithms are memoryless and may achieve excellent performance in dense graphs or even in graphs with $O(n)$ edges, based on Delaunay triangulations, as shown by experimental results of [12, 10]. Unfortunately, these algorithms are not guaranteed to converge. When they fail, one has to use alternative routing algorithms, such as algorithms based on the right-hand rule which are guaranteed to converge as long as the graph is planar. This commutation from greedy to perimeter routing was first proposed in [3] and later explored in a protocol called Greedy-Perimeter Stateless Routing (GPSR) [7]. To extract planar subgraphs from non-planar graphs, RNG, GG or variations of the Delaunay triangulation [19, 5, 2, 13], may be used. As density is important to achieve good routing performance, many authors have focused on increasing it, to create good spanners of $UDG(V)$ [6, 20, 10, 12]. Some of these approaches [6, 10, 12] are based on Delaunay triangulations, because efficient algorithms can be used to build graphs that are good spanners of $UDG(V)$.

Gao *et al.* [6] use a triangulation algorithm that builds a planar graph called *restricted Delaunay graph* (RDG). RDG is a graph that contains $UDel(V)$. Communication cost of their algorithm is $O(n^2)$ In [12], *Li et al.* presented an algorithm that builds $PLDel(V)$ (also a supergraph of $UDel(V)$), with communication cost of $O\left(n \log n\right)$. Lan and Wen-Jing [10] also build $PLDel(V)$ but with higher communication cost ($O(n^2)$). In [13], Li *et al.* presented algorithms that build subgraphs of $UDel(V)$ based on 1 and 2-hop neighbor information. Although their algorithms are simple, their graphs are less dense than any of the previous graphs and it is unclear whether they are good spanners of $UDG(V)$.

Our algorithm improves the results of *Li et al.* [12]. Although the asymptotic communication cost of both algorithms is the same, namely $O(n \log n)$, our algorithm requires one communication step, while [12] requires 4 communication steps. Thus, our algorithm converges much faster. Furthermore, the total signaling cost of our algorithm is much smaller, as we will show in the evaluation section, because in FLDT nodes send only a subset of the Delaunay triangulation in their single communication step (if the subset is empty no message is sent).

4 Triangulation Algorithm

In this section we present a new Fast Localized Delaunay Triangulation (FLDT) algorithm that builds a $PLDel(V)$ graph.

4.1 Description

The FLDT algorithm is decentralized, as it does not rely on any centralized component, and localized, since nodes are only required to gather knowledge about some nodes in their 2-hop neighborhood. The algorithm builds a triangulation that ensures routing between any pair of nodes as long as $UDG(V)$ is connected. The algorithm consists of the following logical steps:

1. The *neighbor discovery* step. The purpose of this step is to allow nodes to discover their neighbors. For sake of clarity, we first describe and analyze the algorithm in the context of a fixed setting, where all nodes know their neighbors *a priori*. The discussion of the use of our algorithm in the context of dynamic settings (that may require the exchange of BEACON messages) is postponed to Section 6.

2. The *triangulation* step. The purpose of this step is to let each node compute and advertise to its neighbors the relevant Delaunay triangulations. Based on the information collected during the neighbor discovery step, each node P locally computes a Delaunay triangulation. For convenience of exposition, we introduce the predicate *Delaunay*$\triangle_P(Q,R)$ that holds true at P if, according to the triangulation computed by node P, triangle $\triangle PQR$ should exist. *Delaunay*$\triangle PQR$ will also be used when referring to the predicate at no particular node. When *Delaunay*$\triangle_P(Q,R)$ holds at P, if $\angle QPR \geq \pi/3$, then P broadcasts a TRIANGULATE $\triangle PQR$ message to all nodes within range.

The purpose of the $\pi/3$ condition is to ensure that no node will issue more than 6 TRIANGULATE messages by its own initiative (as in [12]). Since no additional messages are sent in the following steps, total communication cost of FLDT is $O(n \log n)$. In practice, the constant involved in this bound is small, because, as we show in Section 5, each node announces less than 6 other nodes in average.

3. The *sanity* step. The purpose of this step is to let neighbor nodes eliminate inconsistent Delaunay triangulations. They do so by comparing triangulations computed locally with the triangulations computed by their neighbors in **Step 2**, as advertised by TRIANGULATE messages. Note that by processing TRIANGULATE messages, nodes may learn about new nodes that are not their direct neighbors. This addititional information will never create new Delaunay triangulations, as triangulations must be formed with direct neighbors. However, TRIANGULATE messages may invalidate some of the triangulations computed in **Step 2**. This may happen at P if: i) Q or R broadcast a TRIANGULATE message with some node T that invalidates $\triangle PQR$, i.e., $T \in \bigcirc PQR$, or ii) some node W sends a TRIANGULATE message with an intersecting triangle WXZ, where either X or Z invalidate $\triangle PQR$, i.e., $X \in \bigcirc PQR$ or $Z \in \bigcirc PQR$. Case i) ensures that a node only maintains a predicate if its neighbors are not aware of some node that invalidates it, while case ii) avoids the existence of intersections [1].

4. The *Gabriel edges* step. The purpose of this step is to add to the graph all missing Gabriel edges. Otherwise, despite always being correct, a Gabriel edge PQ for which no predicate *Delaunay*$\triangle_P(Q,R)$ holds at P (e.g., after switching to false in **Step 3**) would not be included by P. This will increase the density of the graph, while keeping $O(n)$ edges (note that a Gabriel edge always belongs to the Delaunay triangulation and can be determined locally without additional exchange of information).

Optimization. To simplify our algorithm, all TRIANGULATE messages should be sent in a single control message. $\square$

When comparing FLDT with previous solutions [12, 10] one must notice that the simplicity of our algorithm comes from two insights, that we later prove correct in Section 4.2. First, proposals sent in TRIANGULATE messages, alone, suffice to confirm or reject triangulations proposed by neighbors in their own TRIANGULATE messages (and vice-versa), i.e., there is no need to dedicated replies. This insight builds on the observation that two Delaunay neighbors do not need to agree on some predicate *Delaunay*$\triangle PQR$. It can hold at P but not at Q and R if these two latter nodes are out of range of each other. The fundamental issue is, in fact, to ensure that two nodes P and Q always agree on whether

[1] Note that case i) can also prevent some intersections.

edge PQ should exist (Lemma 4). Second, if three nodes P, Q and R wrongly assume the existence of $\triangle PQR$, intersected by $\triangle WXZ$, such that one of the nodes of $\triangle WXZ$ is inside $\bigcirc PQR$, then P, Q and R will listen to the same TRIANGULATE message on $\triangle WXZ$, thus commuting the predicate *Delaunay*$\triangle PQR$ to false simultaneously at P, Q, and R (Lemma 5).

4.2 FLDT Builds $PLDel(V)$ in a Single Communication Step

In this Section we show that, after a single communication step, our algorithm builds $PLDel(V)$. To see this, we reason as follows and present the necessary proofs afterward. The triangulation computed at step 2 of the algorithm is a super-graph of $LDel^{(1)}(V)$. Only step 3 of the algorithm removes edges from the graph: either edges from triangles that did not belong to $LDel^{(1)}(V)$ in the first place, and edges from all intersecting triangles that did not belong to $LDel^{(2)}(V)$. Therefore, the graph built by FLDT is a subgraph of $LDel^{(1)}(V)$ (Lemma 3), which is planar (Lemma 5). In fact, this graph is $PLDel(V)$ (Theorem 1).

In the proofs we assume that the network is static and that links are perfect (*i.e.*, no messages are lost). This assumption is made for sake of clarity. In Section 6 we discuss how lossy links can be addressed by the algorithm in practical dynamic settings (where nodes can join or leave). Note also that in the proofs we assume that no four nodes are co-circular (this scenario can be trivially addressed using simple tie-breaking rules).

Lemma 1 *In the $UDG(V)$ model, if two edges AB and CD of a given node set V intersect, then at least one of the nodes is within communication range of the other three.*

Proof 1 *We first note that if AB intersects CD and if $d(A, B)$ includes C, C knows of A, B and D. Since AB intersects CD, $d(A, B)$ and $d(C, D)$ have overlapping areas (one may even contain the other) and hence it follows that at least one of the nodes (e.g., C) is inside the circle defined by the other pair of nodes (e.g. $d(A, B)$), thus proving the Lemma.* □

Lemma 2 *If after the Delaunay triangulation computed at step 2 of the FLDT algorithm, Delaunay $\triangle_A(B, C)$ holds, but edge AB cannot exist at B, B will send a* TRIANGULATE *message with at least one node $D \in \bigcirc ABC$.*

Proof 2 *Refer to Figure 1. Since non-Gabriel edge AB exists at A, C must be inside $d(A, B)$ (e.g. see [10]). In this case, AB cannot exist at B if Delaunay$\triangle_B(X, D)$ holds at B for some nodes X and D and XD intersects AB (assume w.l.o.g. that X and C are on the same side of AB, possibly with $X = C$). $D \in \bigcirc ABC$, because otherwise $\bigcirc BXD$, would contain A which would be a contradiction (D' in the figure must be outside $d(A, B)$ and closer to B than to A: $\bigcirc D'BC$ intersects $\bigcirc ABC$ at B and C, thus for $X = C$ it contains A; if $X \neq C$, $\bigcirc D'BX$ intersects $\bigcirc D'BC$ at B and D', thus containing the part of $\bigcirc D'BC$ that contains A). Since, $\angle XBD > \angle ABD > \pi/3$, B will send information of D in its* TRIANGULATE *messages.* □

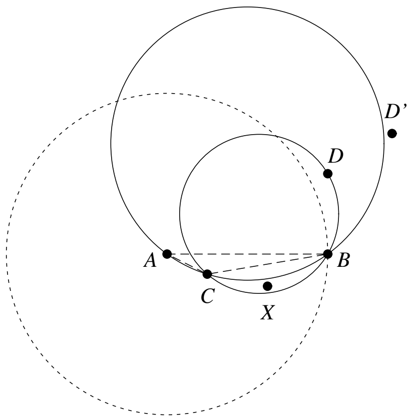


Figure 1: A and B do not agree on $\triangle ABC$

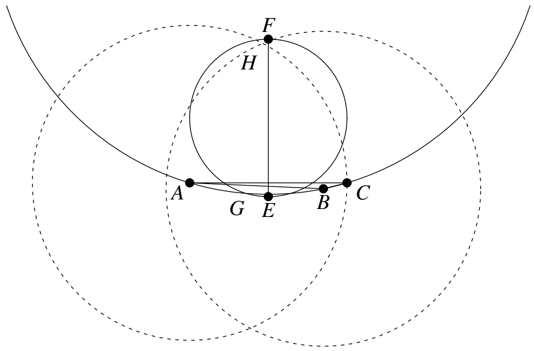


Figure 2: A, B and C wrongly agree on $\triangle ABC$

Corollary 1 *At the end of the FLDT algorithm, any node N, simultaneously neighbor of A and B in the conditions of Lemma 2, will know about some node $D \in \bigcirc ABC$.*

Lemma 3 *At the end of the FLDT algorithm, Delaunay$\triangle_A(B, C)$ holds at A only if there is no direct neighbor D of A, B or C such that $D \in \bigcirc ABC$.*

Proof 3 *If edge AB cannot exist at B, the proof follows from Lemma 2. Hence, we will focus on the case where AB exists at A and B. The case where there is a common neighbor $C \in d(A, B)$ for which Delaunay$\triangle ABC$ holds at A and B, does not contradict the Lemma. Assume now that Delaunay$\triangle_A(B, C)$ holds, while Delaunay$\triangle_B(A, C)$ does not (Delaunay$\triangle_B(A, D)$ holds instead, with C and D lying on the same side of edge AB). Since $C, D \notin d(A, B)$ [10] we can use circumcircles to argue that either $D \in \bigcirc ABC$ with $||DA|| > 1$ or $||BC|| > 1$ (both cases can occur simultaneously). The latter case alone does not contradict the Lemma and, in the former case, since $||AD|| > 1$, $\angle ABD > \pi/3$ and hence, B will send a* TRIANGULATE *message on $\triangle ABD$, thus making A switch Delaunay$\triangle_A(B, C)$ to false. The Lemma follows.* □

Lemma 4 *At the end of the FLDT algorithm, if edge AB exists at A, it must exist at B, either because it is a Gabriel edge or because there is one predicate Delaunay$\triangle ACB$ holding at A and B for some common neighbor $C \in d(A, B)$.*

Proof 4 *Given Lemma 2, the only not so trivial thing to prove is that non-Gabriel edge AB cannot be deleted by A and maintained by B or vice-versa at step 3 of the algorithm. Hence, assume that node A deletes edge AB, because Delaunay$\triangle_A(B, C)$ does not hold anymore due to some intersecting edge with node $D \in \bigcirc ABC$, which is not a direct neighbor of A, B or C. In this case, by Lemmas 1 and 3, A must have received information of D through a common neihbor of A, B and C and Delaunay$\triangle ABC$ will not hold at any of the three nodes A, B or C.* □

As a consequence of the Lemma 3, the final graph is a subgraph of $LDel^{(1)}(V)$ (which may not be planar). The following Lemma serves to ensure that no intersection is possible.

Lemma 5 *Graph built by FLDT is planar.*

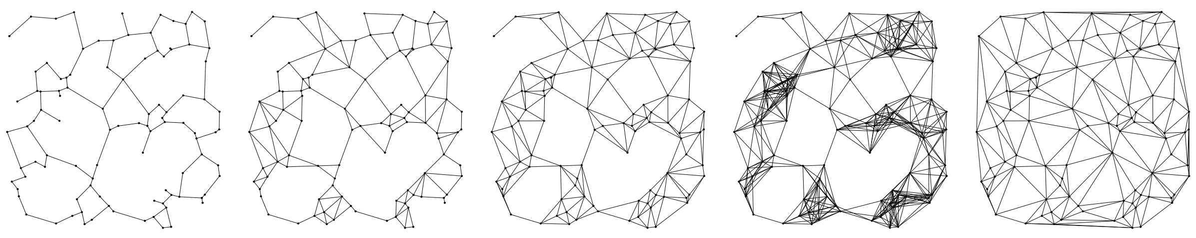
Proof 5 *Refer to Figure 2 [12, 10]. Assume that Delaunay$\triangle ABC$ holds at A, B and C and that $\triangle ABC$ intersects EF (at AB and AC). If E has more than one intersecting edge with AB, assume w.l.o.g. that EF defines the minimum angle $\angle FEA$, with $F \in \bigcirc ABC$ (E cannot define an intersecting edge EF' if $F' \notin \bigcirc ABC$, because, in that case, any circle containing E and F' would have to include at lest one of the nodes A, B or C known by E). By Lemma 4, edge EF exists at E only if EF is a Gabriel edge or if some predicate Delaunay$\triangle EFG$ holds at E and F at the end of the algorithm (assume w.l.o.g. that G is at the left of EF). In the latter case, either EG or GF would also intersect AB and AC. Since by hypothesis F defines the smallest angle $\angle FEA$ it must be GF. By Lemma 1, in this case, G must be within communication range of A, B and C. $\angle AFB < \pi/3 \Rightarrow \angle EFG < \pi/3$, which means that A (the same goes for B and C) will always listen to some* TRIANGULATE *message with edge EF (from E or G) and will eliminate wrong edge AB (AC).*

Now, consider the case where EF is a Gabriel edge. Then, there must be some node G, possibly $G = A$ for which Delaunay$\triangle_E(F,G)$ holds. By hypothesis AB and GE do not intersect. If $\angle FEG > \pi/3$ E sends a TRIANGULATE *message and the Lemma follows. Otherwise, a new subdivision in cases is needed: GE exists at G and GE does not exist at G. In the first case, Delaunay$\triangle_G(E,H)$ holds and H may be, in fact, node F. For reasons similar to the ones given before, $H \in \bigcirc ABC$. $\angle AHB < \pi/3 \Rightarrow \angle GHE < \pi/3$. Since G knows of F, $H \neq F \Rightarrow ||HE|| > 1 \Rightarrow \angle HGE > \pi/3$. This means that either G or E or both will send a* TRIANGULATE *message with information of F or $H \in \bigcirc ABC$. In the second case, if the triangulation computed by G does not include non-Gabriel edge GE then, by Lemma 2, for some $X \in d(E,G)$, G will send information of $H \in \bigcirc EXG \Rightarrow H \in \bigcirc ABC$ above AB. Whether GE exists or not in G, by Lemma 1 and Corollary 1 A, B and C will hear about some intersecting edge with node $H \in \bigcirc ABC$, thus switching Delaunay$\triangle ABC$ to false.* □

We know that since nodes can send their TRIANGULATE messages independently of each other in a single communication step, by Lemmas 3 and 5 and for the reasons explained before, it follows that FLDT builds a subgraph of $PLDel(V)$. However, we still need to prove that it is not possible for some edge $AB \in LDel^{(1)}(V)$ to be incorrectly deleted due to the announcement of some other intersecting edge $EF \notin LDel^{(1)}(V)$.

Theorem 1 *After a single step of communication, FLDT builds the graph $PLDel(V)$.*

Proof 6 *Refer to Figure 2. If $AB \in LDel^{(1)}(V)$ is deleted by existence of edge $EF \notin LDel^{(1)}(V)$ it cannot be a Gabriel edge, because a Gabriel edge is always correct. Hence, $\exists C \in d(A,B)|$ Delaunay$\triangle ABC$ holds at A and B. However, w.l.o.g. $F \in \bigcirc ABC$. Since $EF \notin LDel^{(1)}(V)$, it is not a Gabriel edge and $\exists K_1 \in d(E,F)$ (not shown), such that K_1E or K_1F intersects AB (note that $||K_1E|| < ||EF||$ and $||K_1F|| < ||EF||$). Given that $A, B \notin d(E,F)$ and $A, B \notin d(K_1,E)$ if intersection is with K_1E ($d(K_1,F)$ if intersection is with K_1F), it follows that even if the intersecting*



(a) RNG (b) GG (c) $PLDel$ (d) UDG (e) Delaunay

Figure 3: Example of graphs

edge $\notin LDel^{(1)}(V)$, we can inductively repeat the reasoning until we find one intersecting edge $\in LDel^{(1)}(V)$. Hence, even if AB is deleted due to some edge $EF \notin LDel^{(1)}(V)$, there is some other edge $\in LDel^{(1)}(V)$ that would legitimately delete AB. Theorem follows. $\square$

5 Evaluation

In this section, we compare i) routing performance in each of the following graphs: RNG, GG, $PLDel$, UDG and DT and ii) signaling cost of FLDT versus the algorithm of [12]. Figure 3 illustrates the graphs in a network of 100 nodes. We have used the GPSR routing algorithm [7] in all graphs, except UDG, which is not planar. In UDG we have used the greedy routing algorithm. Results for DT are depicted only to serve as a reference, because, as we have discussed before, such triangulation is not possible in a wireless environment. Since node density has a crucial impact on the performance of routing algorithms, in our experiments, we have distributed a variable number of nodes (between 140 and 600) inside a square of fixed side (7.5 times the communication range). Reader should notice that density cannot be arbitrarily reduced, because disconnected topologies would result with high probability. On the other hand, increasing node density will benefit UDG, because greedy routing will converge with increasingly higher probability and, unlike the remaining graphs, paths will become shorter.

Figure 4 shows the average path length in number of hops (for paths where greedy did not failed), while Figure 5 depicts the percentage of failures for the greedy routing algorithm in the UDG graph. Both curves are functions of the average number of neighbors of a node [2]. From the figures, it is quite evident that when node density is high, no subgraph can do better than UDG, unless memory usage is an issue and a node does not want to maintain all its neighbors. In this case, $PLDel$ may be a good option, because nodes need to maintain only a constant number of neighbors in average. On the other hand, when node density decreases, $PLDel$ is definitely the preferable choice, because it achieves the best performance among the algorithms that ensure routing convergence. Since the

[2]For a node whose communication (unit) disk is entirely inside the simulation square.

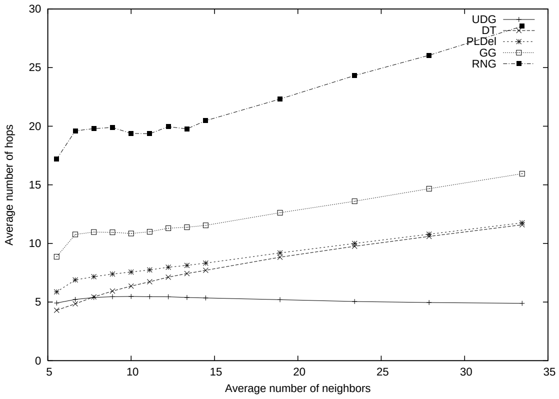


Figure 4: Average number of hops

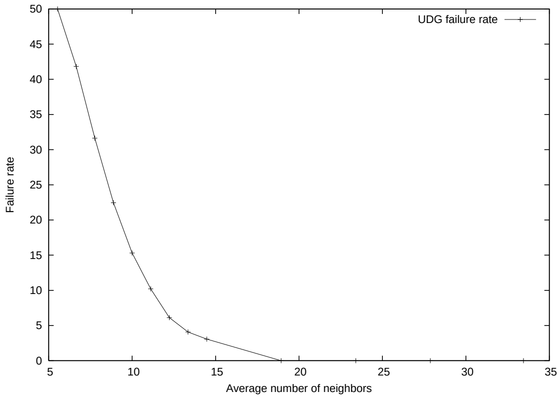


Figure 5: Failure rate in the UDG

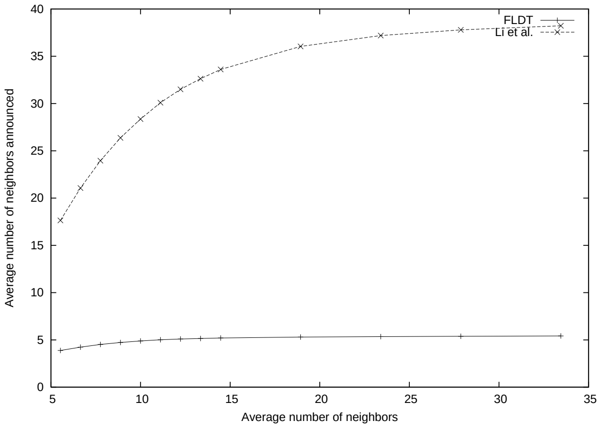


Figure 6: Average number of neighbors announced by each node

possibility of a greedy routing failure always exists, no matter how large node density is, it may also be a good idea to maintain two graphs in memory: UDG and $PLDel$. The point is to use greedy in UDG whenever possible for performance reasons and switch back to a right-hand rule algorithm and to $PLDel$ in case greedy fails. Such solution has the advantage of being oblivious to node density. It is also interesting to observe that the number of hops obtained in $PLDel$ is typically quite close to that number in a DT, for high densities, where all edges are short, but the same is not true when node densities are small, because in these cases, DT uses long edges, thus saving many hops.

To complete our evaluation, we depict in Figure 6 the average number of neighbors announced by each node, in the algorithm of Li *et al.* and in our own algorithm. Note that whenever a triangle is announced, two nodes are counted (the sending node is not counted). The algorithm of Li *et al.* also needs to announce Gabriel edges, which are counted as only one node (again, sending node is not counted). We can see that the number of nodes announced stabilizes in both algorithms as the density increases, and that our algorithm announces approximately between 5.2 and 7 times fewer nodes for the densities of interest. Furthermore, while our algorithm needs a single communication step, the algorithm of Li *et al.* needs 4 steps. Therefore, we believe that these results show that our algorithm builds $PLDel$ very efficiently.

6 Application in Dynamic Settings

So far, we have described the execution of our algorithm in a static setting, where a node knows *a priori* all its neighbors. We now discuss the application of our algorithm in dynamic settings.

The application of any graph building algorithm in a dynamic setting requires a complementary mechanism to discover new nodes and to detect the departure/failure of existing nodes. In an optimized implementation, the concrete mechanisms to be used may depend on the physical and data link layer technology. However, in the literature (for instance, [12, 10]) it is usually assumed that nodes periodically exchange BEACON messages. We would like to emphasize that our algorithm is particularly well suited for such setting, as TRIANGULATE messages can be easily piggybacked to (or even replace) BEACON messages. Therefore, when BEACON messages are required, our algorithm can be implemented with no additional messages, becoming extremely competitive with regard to the Gabriel or the Relative Neighborhood graphs, which are not good spanners of UDG.

Also, for sake of simplicity, we have assumed perfect channels in our exposition (i.e., no message losses). However, in a dynamic setting, BEACON messages have to be exchanged periodically. This means that, at no additional cost in terms of number of messages exchanged, our algorithm may retransmit periodically TRIANGULATE and recalculate $PLDel$ at the end of each period. Therefore, even if links are lossy, it can be shown that, as long as links are fair (*i.e.*, if a message is sent infinitely often by a process p then it can be received infinitely often by its receiver [15]), any new node will eventually participate in the triangulation.

7 Conclusions

Routing protocols for wireless *ad hoc* networks may benefit from using a planar and localized Delaunay triangulation to achieve good routing performance, while, at the same time, guaranteeing convergence. Therefore, in this paper we presented a new algorithm, FLDT, to build a well-known graph called $PLDel$. Our experimental results show that $PLDel$ can be used either to substitute the UDG, when node density is small, or as a complementary graph that ensures routing convergence for all node densities.

FLDT has a communication cost of $O(n \log n)$, which is within a constant of the optimal and requires a single communication step (unlike previous work, that requires 4 communication steps). We have also shown that the signaling cost of FLDT is much smaller than that of previous approaches, due to the small number of control messages. Furthermore, in dynamic settings that require the exchange of beacon messages, our algorithm requires no more messages than the algorithms used to build the very simple but inefficient GG or RNG. Therefore, due to its efficiency, our algorithm is of practical relevance in location-based wireless *ad hoc* networks.

References

[1] Jean-Daniel Boissonnat and Monique Teillaud. On the randomized construction of the Delaunay tree. *Theoretical Computer Science*, 112(2):339–354, 1993.

[2] Prosenjit Bose and Pat Morin. Online routing in triangulations. In *10th Annual Internation Symposium on Algorithms and Computation (ISAAC)*, 1999.

[3] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in *ad hoc* wireless networks. In *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 48–55, 1999.

[4] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, (2):153–174, 1987.

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

[6] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanners for routing in mobile networks. In *2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, 2001.

[7] Brad Karp and H. T. Kung. GPRS: Greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[8] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *11th Canadian Conference on Computation Geometry (CCCG 99)*, 1999.

[9] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'02)*, 2002.

[10] Luan Lan and Hsu Wen-Jing. Localized Delaunay triangulation for topological construction and routing on manets. In *2nd ACM Workshop on Principles of Mobile Computing (POMC'02)*, 2002.

[11] Der-Tsai Lee and Bruce J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242, 1980.

[12] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[13] Xiang-Yang Li, Ivan Stojmenovic, and Yu Wang. Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. *IEEE Transactions on Parallel and Distributed Systems*, 15(4):350–361, April 2004.

[14] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with Delaunay triangulation overlays. Technical Report CS-2001-26, University of Virginia, Department of Computer Science, Charlottesville, VA 22904, 5 2001.

[15] N. Lynch. Distributed algorithms. In *Data Link Protocols*, chapter 16, pages 691–732. Morgan-Kaufmann, 1996.

[16] F. P. Preparata and M. I. Shamos. *Computational geometry: An introduction*. Springer-Verlag, New York, 1985.

[17] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1977.

[18] Ivan Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, July 2002.

[19] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 4(12):261–268, 1980.

[20] Yu Wang and Xiang-Yang Li. Geometric spanners for wireless ad hoc networks. In *The 22nd IEEE International Conference on Distributed Computing Systems*, 2002.