TR 2006/03 ISSN 0874-338X

Single-Step Creation of Localized Delaunay Triangulations

Filipe Araujo
Luís Rodrigues

Centre for Informatics and Systems of the Univesity of Coimbra

Single-Step Creation of Localized Delaunay Triangulations

Filipe ARAUJO
University of Coimbra
filipius@dei.uc.pt
Luís RODRIGUES
University of Lisbon
ler@di.fc.ul.pt

Abstract

A localized Delaunay triangulation owns the following interesting properties for sensor and wireless *ad hoc* networks: 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 two localized algorithms, FLDT1 and FLDT2, that build a graph called planar localized Delaunay triangulation, $PLDel$, known to be a good spanner of the Unit Disk Graph, UDG. Our algorithms improve previous algorithms with similar theoretical bounds in the following aspects: unlike previous work, FLDT1 and FLDT2 build $PLDel$ in a single communication step, maintaining a communication cost of $O(n \log n)$, which is within a constant of the optimal. Additionally, we show that FLDT1 is more robust than previous triangulation algorithms, because it does not require the strict UDG connectivity model to work. The small signaling cost of our algorithms allows us to improve routing performance, by efficiently using the $PLDel$ graph instead of sparser graphs, like the Gabriel or the Relative Neighborhood graphs.

Index Terms

Wireless communication, Routing protocols, Delaunay triangulation

I. INTRODUCTION

In wireless *ad hoc* and sensor networks, nodes typically self-organize and communicate with each other using radio broadcast. Nodes operate on batteries and thus need to run programs with small memory footprints, low CPU requirements and energy-conserving communication protocols. It is therefore utterly important to rely on routing schemes with small state and communication overhead. To meet these

requirements, nodes can run a *localized* routing scheme, where they only need to maintain information about other nodes within a restricted neighborhood. Additionally, for the sake of efficiency, a routing scheme should also be *competitive*, i.e., a path found by the scheme should be at most c times longer than the shortest path.

Position-based routing algorithms usually follow a simple approach to forward packets that takes into account the current position of the node holding the packet and the position of the destination. For instance, in the very simple greedy routing algorithm [8], nodes forward the packet to the neighbor that is closest to destination. Occasionally, packets reach a local minimum node that does not have any available forwarding alternative. One common approach to overcome these minima is to use a right-hand routing algorithm [3], which however needs a planar graph (i.e., without intersection of edges). These algorithms are based on the idea of escaping from a maze by never lifting the right hand from the wall. This form of routing is inefficient, specially in planar graphs with a small number of edges. Hence, this creates a challenge: we want to build a planar graph and yet it should be as dense as possible.

One way of achieving competitive routing with a planar graph is to build a (global) Delaunay triangulation [4]. Unfortunately, building such a graph is not viable in *ad hoc* wireless networks, because: i) edges may be longer than communication range; ii) it cannot be built with local information and therefore, communication cost would be too high. In fact, in general, there is no way of ensuring competitive routing with a localized routing scheme [14]. Despite these obstacles, a *localized* variation of the Delaunay triangulation is an interesting component for position-based routing schemes, because it creates a graph with many edges within the constraint of being planar. In this way, the routing algorithm can achieve a good performance. Our main motivations to use a localized variation of the Delaunay triangulation are precisely the theoretical economy of the triangulation, the performance of the routing algorithm and the fact that all the approaches that we know to create triangulations have some important drawbacks.

In the literature, one can find several algorithms that build Delaunay triangulations for routing purposes, e.g., [17], [18], [19], [10], [15]. All these algorithms have some form of trade-off between the number of

communication steps that they need and the communication cost involved. We define a communication step as the period required for one or more nodes sending causally unrelated messages and for the destination(s) to receive it. As we describe in Section III, the most efficient algorithms that build provably good variants of the Delaunay triangulation may require a single communication step. On the other hand, their communication cost can grow to $O(n^2 \log n)$ bits, as each node may need to advertise each single neighbor that it sees[1]. On the contrary, the most efficient algorithms in terms of asymptotic communication cost require as many as four communication steps, which is too much of a burden in a real wireless *ad hoc* or sensor network. For these reasons, simpler algorithms that create sparser planar graphs (like the Gabriel graph that we review in Section II) are often preferred. These simpler graphs require no communication step besides a beacon message from each node. Nodes use a total communication cost of $O(n \log n)$ bits to send this beacon message.

In this paper, we improve the work of Li *et al.* [17], [18], by presenting two algorithms that are considerably simpler and yet build the same *Planar Localized Delaunay Triangulation* graph ($PLDel$), with the same asymptotic communication cost, but with just a single communication step. We call "Fast Localized Delaunay Triangulation 1" (FLDT1) and "Fast Localized Delaunay Triangulation 2" (FLDT2) to our algorithms. We believe that one of the interesting aspects of our work is that we consider the use of a pair of models that goes beyond the unit dist graph (UDG). In the UDG model, two nodes are neighbors if and only if their distance is at most 1. Despite being unrealistic it is difficult to overcome the simplicity of UDG for position-based routing algorithms, because under this model, designers can be sure that nodes see neighbors located in some specific places and remove the edges that intersect. However, in FLDT1, we do not need a strict unit dist graph (UDG) and nodes are not required to know their communication range. We show in this paper that the minimal graph model where FLDT1 can create connected and planar graphs is tightly related to the Relative Neighborhood Graph and therefore we call it the "Relative Neighborhood Model". The FLDT2 algorithm is an optimization of FLDT1 that (like

[1]We assume that the identification and the position of nodes need $O(\log n)$ bits in an n-node network.

the other aforementioned triangulations) only works in unit disk graphs where nodes are aware of their communication range. This enables FLDT2 to use a very small number of messages, especially in denser networks.

Both FLDT1 and FLDT2 are well suited to wireless environments for the following reasons: i) they are very efficient as they require a single communication step; ii) they are applicable to dynamic and asynchronous settings (see Section VI-B), such as those found in a sensor or mobile network, where nodes can exhaust their batteries or move away from communication range; iii) they are localized, only requiring nodes to receive information broadcast by direct neighbors, thus having a communication cost within a small constant of the optimal (assuming that a beacon message of $O(\log n)$ bits in an n-node network is necessary per node); iv) they require nodes to keep track of only a constant number of neighbors in the average; v) under the constraint of preserving planarity, they build a graph with good density; finally vi) FLDT1 always creates a connected and planar graph in the more realistic Relative Neighborhood Model.

In a previous conference version of this paper [1], we have presented FLDT1 [2]. Here, we present FLDT2, we show that FLDT1 can work under more general models than UDG and, finally, we compare $PLDel(V)$ with an additional triangulation, called "Partial Delaunay Triangulation" [19].

The rest of the paper is structured as follows. For self-containment, we provide a short overview of the necessary background concepts in Section II. In Section III, we provide a survey on related work on Delaunay triangulations in wireless networks. In Section IV, we describe our algorithms and prove their correctness. In Section V we show that FLDT1 can operate under more general connectivity models. In Section VI, we experimentally evaluate our algorithms. Section VII concludes the paper and points direction for future work. In the Appendix we compare the performance of several routing subgraphs that are relevant to our work.

[2]In that paper, we did not need to distinguish this algorithm from another one and therefore, we called it simply FLDT.

II. PRELIMINARIES

A. Notation

Throughout this paper, we will use the following conventions for notation: a triangle defined by nodes A, B and C is denoted $\triangle ABC$; an angle ($< \pi$) between edges AB and AC defined at A is denoted $\angle BAC$ or $\angle CAB$; the disk whose diameter is defined by two nodes A and B is denoted $d(A, B)$; the circumcircle of nodes A, B and C is denoted $\bigcirc ABC$.

B. Initial Graph Model

We assume that nodes can determine their own position either with a GPS-like receiver or with some other alternative mechanism. We also assume that nodes can determine the position of their neighbors, usually through the exchange of beacon messages. Given a set of nodes V in a two-dimensional space, the unit disk graph $UDG(V)$ 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. The set of k-neighbors of node A is denoted $N_k(A)$, while for the special case $k = 1$, this is simply denoted $N(A)$. Of the triangulations that we present in this paper, only FLDT1 does not strictly assume the UDG model.

C. Spanning-ratio

Given an initial graph G and a subgraph H, we say that H is a "t-spanner of G" if and only if there is a constant t such that:

$$\max_{\forall S,D \in V} \left\{ \frac{\|\Pi_H(S,D)\|}{\|\Pi_G(S,D)\|} \right\} \leq t$$

where S and D are respectively the source and destination of a path and $||\Pi_H(S,D)||$ is the length of the shortest path between S and D in graph H. This means that for all nodes S and D, the shortest

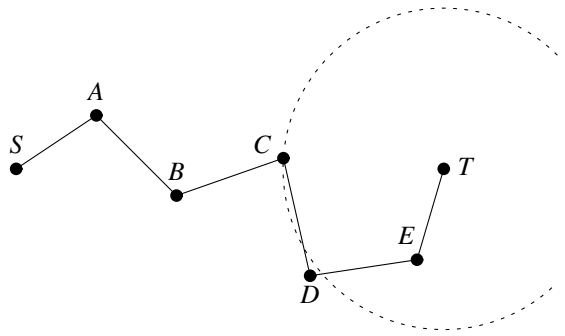


Fig. 1. Greedy Perimeter Stateless Routing

path between S and D in H, $\Pi_H(S, D)$, is at most t times longer than in G, $\Pi_G(S, D)$, where t is known as the "length stretch factor". When the graph G is the complete Euclidean graph determined by V, the above expression defines an "Euclidean t-spanner". In a sense, this factor indicates the quality of the subgraph. The smaller t is, the better the subgraph is and the more likely it is that a packet will use shorter routes (which translates to either fewer and/or shorter hops).

D. Localized Routing Schemes

A routing scheme is comprised of two parts: i) a pre-processing algorithm that prepares data structures needed to support routing decisions (e.g., routing tables or a subgraph of the initial connection graph), and ii) a distributed routing algorithm running at each node that determines the next hop for a message. Given the frequent topology changes and the limited resources available in wireless *ad hoc* networks, it is very convenient to limit the control information used for routing to the minimum possible. Localized routing schemes [12], [5], [17], [11] address this goal. A routing scheme is localized if i) nodes only collect information of neighbors that are at most at a constant number of hops away; ii) in addition, nodes are required to store information of at most a constant number of other nodes and iii) messages can only store information of a constant number of nodes, like the origin and the destination.

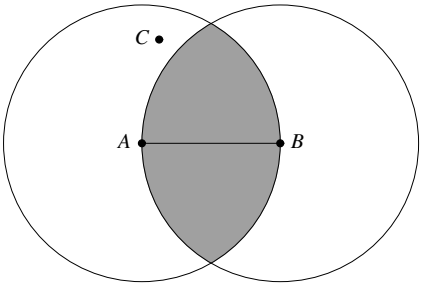


Fig. 2. An edge of the Relative Neighborhood Graph

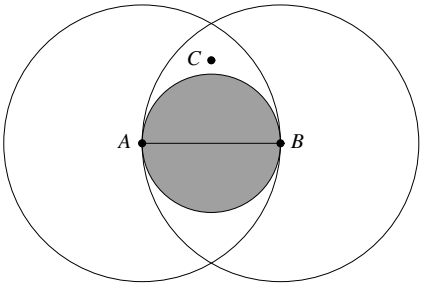


Fig. 3. An edge of the Gabriel Graph

E. Position-based Routing Schemes

Localized routing does not come without costs, because localized knowledge can cause message delivery to fail. One of the simplest ways to ensure the message convergence is to use an algorithm based on the right-hand rule [3]. However, algorithms based on the right-hand rule, also called face or perimeter algorithms, can only ensure convergence in planar graphs (such as FACE-1 [12] and FACE-2 [5]). Furthermore, these algorithms tend to have poor performance. Therefore, to improve routing performance, the right-hand rule can be combined with other approaches, like the Greedy algorithm [8], to create hybrid algorithms such as GFG [5], GPSR [11] (very similar to GFG) or GOAFR+ [13]. In our experiments, we will use the GPSR routing algorithm, because it is simple, it ensures convergence and yet it does not compromise routing performance in most cases. When possible, GPSR uses the greedy strategy of forwarding messages to the neighbor closest to destination. When it finds a local minimum, GPSR switches to perimeter mode and routes around faces. As soon as it finds a node closest to destination than the previous local minimum, GPSR goes back to greedy mode. Figure 1 illustrates the idea. Routing from S to T uses greedy routing in the links $S-A-B-C$ and $D-E-T$. However, since node C sees itself as a local minimum with respect to T, it must use the perimeter mode to route to D.

F. Basic Planar Subgraphs

The initial wireless connection graph, e.g., UDG, is typically not planar, specially in denser networks. Hence, one of the most fundamental problems of single-path position-based schemes is the creation of

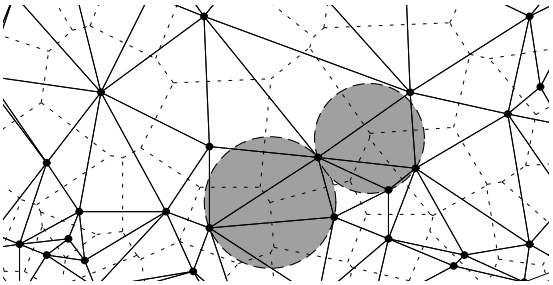


Fig. 4. Delaunay Triangulation, Voronoi tessellation and the emtpy circumcircle property

a planar subgraph from the initial graph (be it UDG or other), such that GPSR (or a similar algorithm) is guaranteed to converge. This is the role of the pre-processing algorithm. Among the simplest planar graphs, we have the "Relative Neighborhood Graph" (RNG) and the "Gabriel Graph" (GG). One of the strongest points of these graphs is that nodes can create their own local views using only position and identification of their neighbors.

The 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). In Figure 2, edge AB is a RNG edge if and only if the gray area is empty of nodes. The GG is comprised of all edges AB such that $d(A, B)$ does not contain any other node of V. This is represented in Figure 3, where the gray area must be empty of nodes. The edges of the GG are called *Gabriel edges*. It should be noted that RNG is a subgraph of GG. While using few resources to create a planar graph is important, it is only part of the problem. It is also desirable to create good spanners of the initial connection graph. Unfortunately, this is not the case of RNG and GG, which are bad spanners of UDG: their use seriously compromises routing performance [7]. Appendix A includes additional experimental data that confirms this result.

A denser planar graph is the "Delaunay triangulation" (DT) of a node set V, also represented as $Del(V)$. $Del(V)$ 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 the Delaunay triangulation, known as the "empty circumcircle" property, states

that the circumcircle of a triangle does not contain any node of V. The Delaunay triangulation has an associated dual concept called, the "Voronoi tessellation". The Voronoi tessellation partitions the space into convex polytopes in the following way. Given a node set V, the polyhedron of node N is comprised of the points that are closer to N than to any other node of V. Figure 4 illustrates the relation between the Voronoi tessellation and the Delaunay triangulation: two nodes share a Delaunay edge if and only if their Voronoi cells have a common border. We can also see the empty circumcircle property for two of the triangles, as no fourth node is inside the gray circumcircles. The DT is a supergraph of the GG and, consequently of the RNG. The Delaunay triangulation naturally emerges as a good subgraph, because it is a $\frac{1+\sqrt{5}}{2}\pi$-spanner of the complete Euclidean graph[3] [6] and can be computed in a deterministic way if nodes share the same information. In [20], Liebeherr *et al.* proposed an algorithm to build a Delaunay triangulation that serves as an overlay network on top of IP. Unfortunately a Delaunay triangulation is not suitable for wireless environments for two main reasons: i) it may have edges longer than 1; and ii) it cannot be built by a localized algorithm. In fact, ensuring the empty circumcircle property may require knowledge of nodes that may be arbitrarily distant in terms of hops, even if physically close. For these reasons, most triangulations used in wireless environments are variations of the Delaunay triangulation. Some of the triangulations that we present in the next section are still good spanners of UDG without incurring in the problems of a complete Delaunay triangulation.

III. Related Work

Under the UDG model, a complete Delaunay triangulation may not exist, because some edges may be longer than 1. As a result of this impossibility, many variations of the Delaunay triangulation have been proposed in the recent past. Perhaps the most obvious variation is the Unit Delaunay triangulation, defined as $UDel(V) = Del(V) \cap UDG(V)$, which is a $(4\sqrt{3}\pi)/9$-spanner of $UDG(V)$ [17]. Gao *et al.* define the *Restricted Delaunay Graph*, RDG, as any planar subgraph that contains $UDel(V)$. Of central importance to us, in this paper, is the definition proposed by Li *et al.* in [17] of *k-localized Delaunay*

[3]This bound was later improved to $(4\sqrt{3}\pi)/9$ [17].

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.* [17] proved that $LDel^{(k)}(V)$ is planar for $k \geq 2$, but edges may intersect for $k = 1$. Unfortunately $LDel^{(2)}(V)$ is more difficult to create than $LDel^{(1)}(V)$. For this reason, Li *et al.* proposed the graph $PLDel(V)$ in [17] and used a slightly different definition in a later work [18]. Lan and Wen-Jing also create a similar graph in [15]. Hence, we assume a broad definition of $PLDel(V)$ to be a planar subgraph of $LDel^{(1)}(V)$, which is also a super-graph of $LDel^{(2)}(V)$. Note that both $LDel^{(k)}(V)$ and RDG are super-graphs of $UDel(V)$, i.e., $RDG \supseteq UDel(V)$ and $LDel^{(k)}(V) \supseteq UDel(V)$. Hence RDG, $PLDel(V)$ and $LDel^{(k)}(V)$, for all k, are $(4\sqrt{3}\pi)/9$-spanners of $UDG(V)$.

The reader should notice that the communication cost to build the triangulations may vary. Gao *et al.* [10] need a communication cost of $O(n^2 \log n)$ to build an RDG. Lan and Wen-Jing [15] also need a communication cost of $O(n^2 \log n)$ to build $PLDel(V)$. Only the algorithm of *Li et al.* [17], [18] has an optimal communication cost of $O(n \log n)$. However, this algorithm needs four communication steps to converge.

The Partial Delaunay Triangulation, PDT, introduced by Li *et al.* in [19], only requires knowledge in direct neighbors. The PDT graph includes all Gabriel edges as well as the following ones. If $d(A, B)$ contains nodes in both sides of edge AB then, $AB \notin Del(V)$ and therefore A deletes it. Hence, consider that $d(A, B)$ only contains nodes in one side of edge AB. There must be some node C that maximizes $\angle ACB$. Let $\alpha = \angle ACB$. A can add edge AB if i) $\bigcirc ACB$ is empty of nodes from $N(A)$ and ii) $\sin \alpha > \frac{d}{R}$, where R is the transmission range of the nodes and $d = ||AB||$. The rationale for this is that if the diameter of the circumcircle $\bigcirc ABC$ is not greater than the communication range of node A and B, A (and B) may be certain that the circumcircle is empty of nodes (the reader is referred to

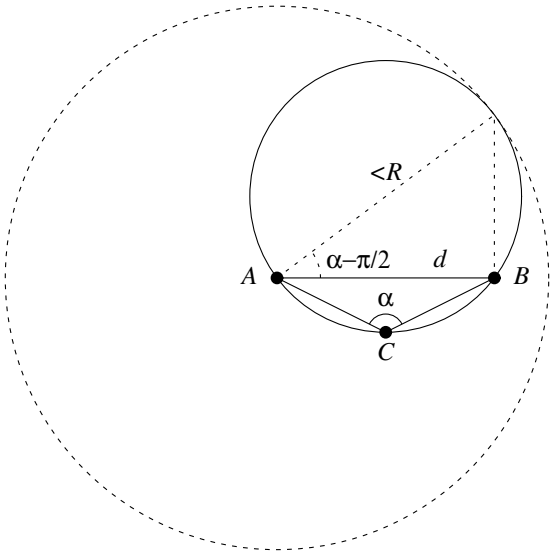


Fig. 5. How to determine if edge AB belongs to the PDT graph

Figure 5). Although this algorithm is simple and requires no additional communication step after neighbors are known, the PDT graph is less dense than any of the aforementioned triangulations. In Figure 6, we illustrate the relations of inclusion between the graphs presented before (in this figure, a graph of a given layer includes all the graphs of the underlying layers; as a result, density increases from the bottom to the top). The graphs on top do not fulfill our needs, because they are not planar. The double lines depict the limits of the *planar* variants of the Delaunay triangulation that are super-graphs of $UDel(V)$. All the graphs that are between those lines are $(4\sqrt{3}\pi)/9$-spanners of $UDG(V)$.

To compute any of the former triangulations, nodes also need an algorithm to compute the Delaunay triangulation of a point set. In literature, we can find several algorithms that build Delaunay triangulations, e.g. [16], [9], [22]. Of particular interest to us are the algorithms that allow Delaunay triangulations to be computed in an incremental way [2], [23], as new nodes that arrive later do not force a recomputation of the entire triangulation.

IV. Triangulation Algorithms

In this section, we present the Fast Localized Delaunay Triangulation algorithms (FLDT1 and FLDT2) that create the $PLDel(V)$ graph. Our algorithms improve the results of *Li et al.* [17], [18]. Although

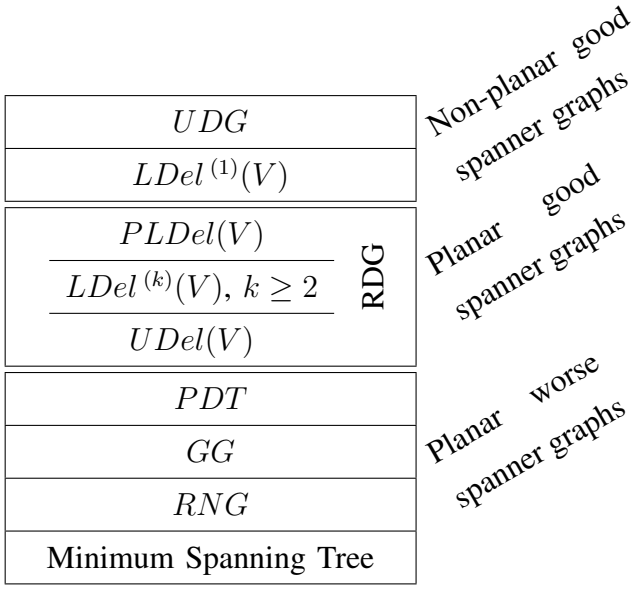


Fig. 6. Relation between some well-known graphs

the asymptotic communication cost of all these algorithms is $O(n \log n)$, our algorithms require one communication step, while [17] requires 4 communication steps. In fact, our algorithms are optimal in the sense that given the knowledge of direct neighbors, at least one communication step is needed to create a super-graph of $UDel(V)$. Furthermore, the total signaling cost of both our algorithms is much smaller than that of previous ones, as we will show in the evaluation section. The reason for this is that, in FLDT1 and FLDT2, nodes send only a subset of their Delaunay triangulation in a single communication step (if the subset is empty no message is sent).

Although FLDT1 uses more messages than FLDT2, it does not require nodes to know their communication range. Such knowledge is required by the algorithm of *Li et al.* [17], [18], PDT, and FLDT2. Additionally, we will show that FLDT1 resists to graph models that extend the UDG model. We believe that these are considerable advantages in practical settings.

A. Description of FLDT1

The FLDT1 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 VI-B.

2. The *triangulation* step. The purpose of this step is to let each node compute and advertise the relevant Delaunay triangulations to its neighbors. 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 [17]). Since no additional messages are sent in the following steps, total communication cost of FLDT1 is $O(n \log n)$. In practice, the constant involved in this bound is small, because, as we show in Section VI, 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[4].

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 step 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. □

When comparing FLDT1 with previous solutions [17], [15] one must notice that the simplicity of our algorithm comes from two insights, that we later prove correct in Section IV-C. 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 edge PQ should exist (Lemma 5). 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 10).

B. Description of FLDT2

The FLDT2 algorithm differs from FLDT1 in step 2. In FLDT1, a node P announces all triangles PQR for which *Delaunay*$\triangle_P(Q, R)$ holds at P and $\angle QPR \geq \pi/3$. However if all the nodes of the triangle

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

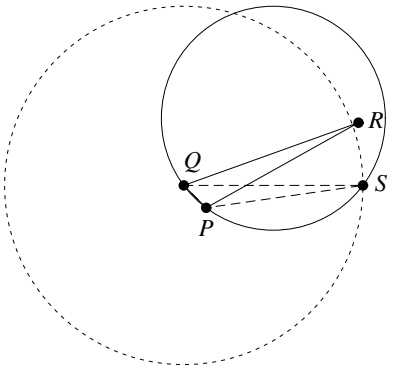


Fig. 7. Edge announced by FLDT2

are aware of each other, announcing the triangle works like a kind of positive acknowledgment from the announcing node. In many triangles, this is pointless (because the three nodes agree on the triangle). It is better to let one of the nodes, say P, announce a node that rejects the triangle if P is aware of such a node. In a sense this will implicitly work as a negative acknowledgment for the triangle. P only announces triangles when the other two neighbors Q and R are *not* aware of each other, i.e., where besides the previously stated conditions, $||QR|| > 1$. This prevents Q and R from creating wrong triangulations (possibly with P). Figure 7 gives an example where node P needs to announce $\triangle PQR$. Since Q is not aware of R, it is assuming that $\triangle QPS$ is correct. However, it will switch predicate *Delaunay* $\triangle_Q(P, S)$ to false after receiving the announcement from P. On the other hand, *Delaunay* $\triangle PRS$ holds at the three nodes and all the edges are short. This makes any announcement of this triangle unnecessary. Hence, we only change step 2 of FLDT1 to:

2. The *triangulation* step. When *Delaunay*$\triangle_P(Q, R)$ holds at P, if $\angle QPR \geq \pi/3$ **and** $||QR|| > 1$, then P broadcasts a TRIANGULATE $\triangle PQR$ message to all nodes within range.

In FLDT2, nodes need to infer whether or not two of their neighbors are within range of each other. This limits the use of FLDT2 to scenarios where nodes are aware of their communication range (algorithms like the one of *Li et al.* [17], [18] and PDT also have similar limitations). Nevertheless, FLDT2 is an optimization of FLDT1 that saves many messages in a strict UDG model.

C. In the UDG Model, FLDT1 and FLDT2 Create $PLDel(V)$ in a Single Communication Step

From the algorithms, it follows that nodes running FLDT1 and FLDT2 use a single communication step. Hence, in this section we need to prove that these algorithms build, in fact, the graph $PLDel(V)$. In Lemma 6, we show that if non-Gabriel edge AB exists at A, there must be some $C \in d(A, B)$ such that *Delaunay* $\triangle ABC$ holds at the three nodes. This result stands on top of Lemmas 1 to 5 and we use it to prove, in Lemma 9, that intersections are impossible at the end of step 3 of the algorithms. From this point, we prove in Lemma 10 that no intersection is possible at the end of the algorithms and in Lemma 11 that the final graph is a planar subgraph of $LDel^{(1)}(V)$. Our final result is Theorem 1, which proves that we build, in fact, $PLDel(V)$. We state all the proofs for the FLDT2 algorithm, but they can be trivially extended for FLDT1. One crucial aspect here is that most Lemmas also hold for FLDT1 in other models beyond UDG. In this way, we can adapt demonstrations to other more general models (see Section V). In all the proofs, we assume that there are no four co-circular nodes. Simple tie-breaking mechanisms can remove co-circularities, if they ever occur in practice.

Lemma 1: If two edges AB and XY intersect, then at least one of the nodes is within communication range of the other three.

Lemma 2: Consider any circumference going through A and B and assume that edge XY intersects edge AB, such that X and Y are both outside $d(A, B)$. There is no circle going through X and Y that does not include either A or B or both.

Lemma 3: If, at the end of the step 2 of the FLDT2 algorithm, non-Gabriel edge AB exists at A and B, there must be some node C, such that $C \in d(A, B)$ maximizes $\angle ACB$ and *Delaunay* $\triangle ABC$ holds at A and B.

Corollary 1: If, at the end of step 2 of FLDT2, non-Gabriel edge AB exists at A, there must be some node C, such that $C \in d(A, B)$ maximizes $\angle ACB$ and *Delaunay* $\triangle_A(B, C)$ holds.

The proofs of these lemmas and of Corollary 1 are either straightforward or known results (see, for instance [10] and [15]).

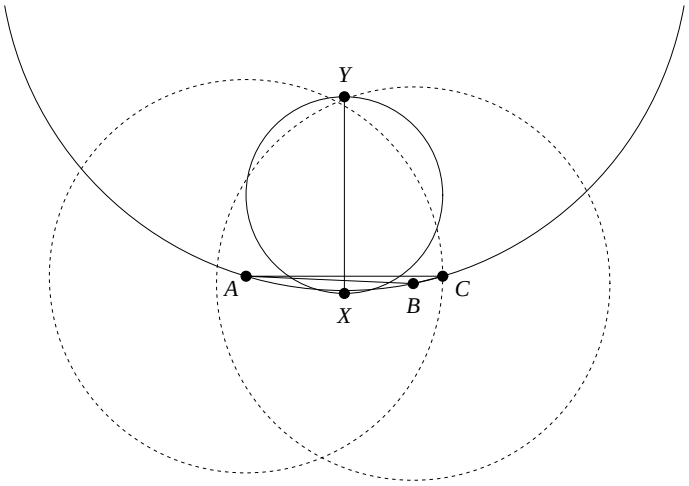


Fig. 8. Possible intersection

Lemma 4: If, at the end of step 2 of FLDT2, *Delaunay* $\triangle_A(B,C)$ holds, but edge AB does not exist at B, then B must have sent a TRIANGULATE message including some node D such that $D \in \bigcirc ABC$ and CD intersects AB.

Proof: AB is not a Gabriel edge. From Corollary 1, if non-Gabriel edge AB exists at A, there is some node $C \in d(A,B)$ such that *Delaunay* $\triangle_A(B,C)$ holds. If AB does not exist at B there must be some X and D such that XD intersects AB and *Delaunay*$\triangle_B(X,D)$ holds at B. We can assume without loss of generality that X and C are on the same side with respect to AB and possibly $X = C$. Since $C \in d(A,B)$, $\bigcirc ABC \subset d(A,B)$ in the same side of AB as X and C are, i.e., all the circumcircle $\bigcirc ABC$ in the same side of X and C is visible to A. $C \in d(A,B)$ is visible to B. Hence, either $X = C$ or $X \notin \bigcirc ABC$ which, from Lemma 2 and from the definition of Delaunay triangle, implies that $D \in \bigcirc ABC$, otherwise, XD could not be a Delaunay edge. Since, $\angle XBD > \angle ABD > \pi/3$ and $||AD|| > 1$, B must have sent information of D in its TRIANGULATE messages. ■

Lemma 5: If at the end of the FLDT2 algorithm, edge AB exists at A it also exists at B.

Proof: If AB is a Gabriel edge, step 4 ensures that it exists at A and B. If it is a non-Gabriel edge and it exists at one of the nodes, but not in the other, it follows directly from Lemma 4 that the node that has it must delete the edge AB at step 3 of FLDT2. Now assume that both nodes A and B agreed

on the edge at the end of step 2. From Lemma 3 there is a single node C such that *Delaunay* $\triangle ABC$ holds at A and B. It could occur that one of the nodes, say A deleted the edge AB due to a message sent by some node X, not heard by B, announcing a triangle with some node $Y \in \bigcirc ABC$ such that XY intersected $\triangle ABC$. In this case, XY must intersect two edges of $\triangle ABC$ and since A and B are not aware of Y, from Lemma 1 they must be both aware of X. Therefore, B and C would also have to listen to the message of X, thus contradicting the initial assumption. ■

Lemma 6: If, at the end of the FLDT2 algorithm, non-Gabriel edge AB exists at A, there must be some third node C, such that $C \in d(A, B)$ maximizes $\angle ACB$ and *Delaunay* $\triangle ABC$ holds at A, B and C.

Proof: If *Delaunay* $\triangle ABC$ holds at A, from Lemmas 3 and 5, it must also hold at B and we know that C maximizes $\angle ACB$. Since $C \in d(A, B)$, $\angle ACB > \pi/2 > \pi/3$. Now assume that *Delaunay* $\triangle_C(A, B)$ does not hold, because C is aware of at least one node inside $\bigcirc ABC$. In this case, two possibilities exist: either C includes AC in its triangulation or it does not. If it does not, from Lemma 5 it follows that *Delaunay* $\triangle_A(B, C)$ could not hold at the end of the algorithm. Therefore, assume that AC exists (so does BC). In this case, there must be some node $D \neq B$, such that *Delaunay* $\triangle_C(A, D)$ holds and CD intersects AB. $D \in \bigcirc ABC$, or otherwise, $\bigcirc ACD$ would contain B, which would be a contradiction. Since $||AD|| > 1$ and $\angle ACD > \pi/3$, C would announce this triangle which would make A switch *Delaunay* $\triangle_A(B, C)$ to false. The same reasoning could be made about B and the Lemma follows. ■

Lemma 7: If, at the end of the FLDT2 algorithm, XY intersects AB and Y is not aware of A and B (and vice-versa), then there must be some node $C \in d(A, B)$ such that *Delaunay* $\triangle ABC$ holds at A, B and C and $Y \in \bigcirc ABC$.

Proof: From Lemma 5, AB must exist at A and B and XY must exist at X and Y. Since A and B are not aware of Y (and vice-versa), from Lemma 1, we immediately know that X is within range of the other three nodes. Since X is aware of A, B and Y, edge AB must not be a Delaunay edge and

consequently it is not a Gabriel edge. Then, from Lemma 6, there must be some node $C \in d(A, B)$ such that *Delaunay* $\triangle ABC$ holds at A, B, C. Clearly $X \notin \bigcirc ABC$, because A and B are aware of X. Therefore, from Lemma 2, $Y \in \bigcirc ABC$, because X considers XY to be a Delaunay edge. ■

Lemma 8: If at the end of the FLDT2 algorithm, edges AB and XY intersect, then for one of the nodes, say Y, we have that Y is aware of X, but not of A or B.

Proof: From Lemma 1, one of the nodes is within range of the other three. To satisfy our hypothesis, this could only be X. From Lemma 5, AB must exist at A and B and XY must exist at X and Y. Assume that Y is in range of A. In this case, either A could not consider edge AB to be a Delaunay edge, or X could not consider XY to be a Delaunay edge, because a Delaunay triangulation cannot have intersections. Therefore, we must conclude that $||AY|| > 1$ and $||BY|| > 1$, because by hypothesis and from Lemma 1 X is aware of A, B and Y. ■

Lemma 9: If two edges AB and XY intersect at the end of the step 2 of the FLDT2 algorithm, at least one of them is deleted in the step 3 of the algorithm.

Proof: Assume that the intersection persists at the end of step 3 of the FLDT2 algorithm. This means that it will also exist at the end of the algorithm and from Lemmas 7 and 8, we can assume without loss of generality that i) $||YA|| > 1$ and $||YB|| > 1$ and ii) there is some node $C \in d(A, B)$ such that *Delaunay* $\triangle ABC$ holds at A, B and C at the end of the FLDT2 algorithm and $Y \in \bigcirc ABC$. Also, from Lemma 2, $X \in d(A, B)$ and it must be on the same side of AB as C is, otherwise AB could never exist. However, from Lemma 6, $C \neq X$ and either AC intersects XY or BC does. We can assume without loss of generality that it is AC. Since X is aware of A, C and Y, AC is not a Gabriel edge (otherwise, XY would not exist at X). Hence, AC is in the same conditions as AB of the hypothesis of this Lemma. The only difference is that AC intersects XY closer to X than AB does. This means that for any intersecting edge we can find another intersecting edge different from any of the previous. This is a contradiction, because the number of nodes is finite. The Lemma follows. ■

Corollary 2: If at the end of step 2 of the FLDT2 algorithm, Gabriel edge XY intersects non-Gabriel

edge AB, AB must be deleted at step 3.

At this point, we still need to prove that the last step of the FLDT2 algorithm (the Gabriel edges step) cannot create a new intersection.

Lemma 10: At the end of the FLDT2 algorithm, there can be no intersections.

Proof: From Lemma 9, there can only be an intersection between AB and XY if one of them (a Gabriel edge) is created at step 4 of the FLDT2 algorithm. Assume without loss of generality that XY is the Gabriel edge. Also, assume that XY is created for the first time at step 4. Clearly, edge AB is not a Gabriel edge and from Lemma 2, either X or $Y \in d(A, B)$ (assume without loss of generality that it is X). This means that the triangulation of X must have included at least two triangles with XY implying that it must have existed at the end of step 2, thus contradicting our initial assumption. Now, assume that edge XY was deleted at step 3 and re-added at step 4. From Corollary 2, in this case, edge AB must have been deleted at step 3 and there will be no intersection. ■

Lemma 11: In the UDG model, FLDT2 creates a subgraph of $LDel^{(1)}(V)$ without intersections.

Proof: An edge AB that exists in the final graph must either be a Gabriel edge or an edge for which there is some node $C \in d(A, B)$ such that ***Delaunay*** $\triangle ABC$ holds at A, B and C (Lemma 6). This means that the final graph is a subgraph of $LDel^{(1)}(V)$. Since, by Lemma 10, there can be no intersections, the Lemma follows. ■

Theorem 1: In the UDG model, FLDT2 builds $PLDel(V)$.

Proof: First, we note that if $\forall K \in \bigcirc ABC, K \notin N_2(A), K \notin N_2(B) \wedge K \notin N_2(C)$, ***Delaunay*** $\triangle ABC$ will hold at A, B and C. From the definition of $LDel^{(k)}(V)$, this means that the final graph is a supergraph of $LDel^{(2)}(V)$. Therefore, assuming the Definition of $PLDel(V)$ of Section III, this Lemma follows from the fact that we build a planar subgraph of $LDel^{(1)}(V)$ (Lemma 11). ■

V. A More General Graph Model for FLDT1

A. The Relative Neighborhood Model and the Gabriel Model

Until now, we have assumed that FLDT1 operates in the UDG model and, in fact, FLDT1 strictly requires the UDG model to create $PLDel(V)$. However, FLDT1 can create a connected and planar graph under more general (and therefore more realistic) graph models. In other words, although we cannot ensure construction of $PLDel(V)$ in more realistic settings, we can nevertheless create a graph where the GPSR algorithm will converge. Hence, one interesting question is to know in which kind of models — less demanding than UDG — does FLDT1 work.

We will start by a model that is strongly related to the RNG. Therefore, we will call this the "Relative Neighborhood Model" (RNM). In the RNM, if node A is aware of node B then, both A and B are aware of any node C if $||AC|| < ||AB||$ and $||BC|| < ||AB||$. An equivalent way of defining this model is to consider that given a triangle $\triangle ABC$, if the two nodes of the longest edge, say A and B are aware of each other, then, the three nodes are all aware of each other. The "visibility zone" of nodes A and B corresponds to the gray area of Figure 2. We shall prove that FLDT1 works under the RNM.

We will also consider a model related to the GG, to which we call "Gabriel Model" (GM). In the GM, if node A is aware of node B then, both nodes A and B are aware of any node C such that $C \in d(A, B)$. The "visibility zone" of nodes A and B corresponds to the gray area of Figure 3. Although FLDT1 may not work under the GM, we are interested in this model, because it is trivial to show that algorithms to create both the GG and the RNG also work under this model (although the RNG algorithm may not create exactly the RNG).

One of the most interesting aspects of these models is that they do not assume that all nodes have the same predetermined communication range. The range of a node can vary, even with the direction of communication. In fact, the UDG model is contained in the RNM and the RNM is contained in the GM, which is the most general of the three. The goal of these models is to define *minimal* communication conditions that ensure proper operation of FLDT1. In other words, we are interested in showing that

FLDT1 works even if the UDG model does not hold for some nodes, but the RNM does. If we get less than the RNM, the network may have intersections.

B. FLDT1 Creates a Planar Connected Graph Under the RNM

In the GM, it is possible to find a counter-example where the FLDT1 algorithm creates a graph with intersections. Nevertheless, FLDT1 works in the RNM. The demonstration for this is a variation of the similar proofs for the UDG model. Therefore, we omit the parts of this proof that are most similar to UDG and only include the most relevant differences, because the remaining parts are easy to derive. One of the most significant differences of these proofs, when compared with the UDG model is that here, in Figure 8, $\angle AYB$ may be greater or equal than $\pi/3$.

Lemma 12: Consider the triangle ABC. Assume that A is aware of B and C and that B and C are not aware of each other. In the RNM, this can only occur if $\angle CAB \geq \pi/3$.

Proof: Assume that $\angle CAB < \pi/3$. In this case, either AB or AC is the longest edge of the triangle. Since A is aware of B and C, this is a contradiction, because the RNM ensures that B and C must be aware of each other. ■

Lemma 12 is fundamental to prove that FLDT1 works under RNM. To understand why, consider the FLDT2 algorithm in the UDG model and assume that *Delaunay* $\triangle_A(B, C)$ holds, with $||BC|| > 1$. In this case, $\angle BAC > \pi/3$ and A will announce the triangle $\triangle ABC$. In the FLDT1 algorithm under RNM, we have a similar situation: if *Delaunay* $\triangle_A(B, C)$ holds, but B and C are not aware of each other, by Lemma 12 we know that $\angle BAC > \pi/3$ and that A will *also* announce the triangle $\triangle ABC$, just in the FLDT2/UDG case. This similarity is crucial for FLDT1 under the RNM and to the result that we state in Theorem 2, which is the RNM counterpart of Lemma 11 for UDG. Given the conclusions of Lemma 12, we omit the proof of this theorem, as this is straightforward given the FLDT2/UDG case.

Theorem 2: In the RNM, FLDT1 creates a subgraph of $LDel^{(1)}(V)$ without intersections.

Under the light of this theorem we reason as follows: we should use FLDT1 instead of the GG, because it is a much better spanner under the ideal UDG. However, even if the communication conditions deteriorate,

TABLE I

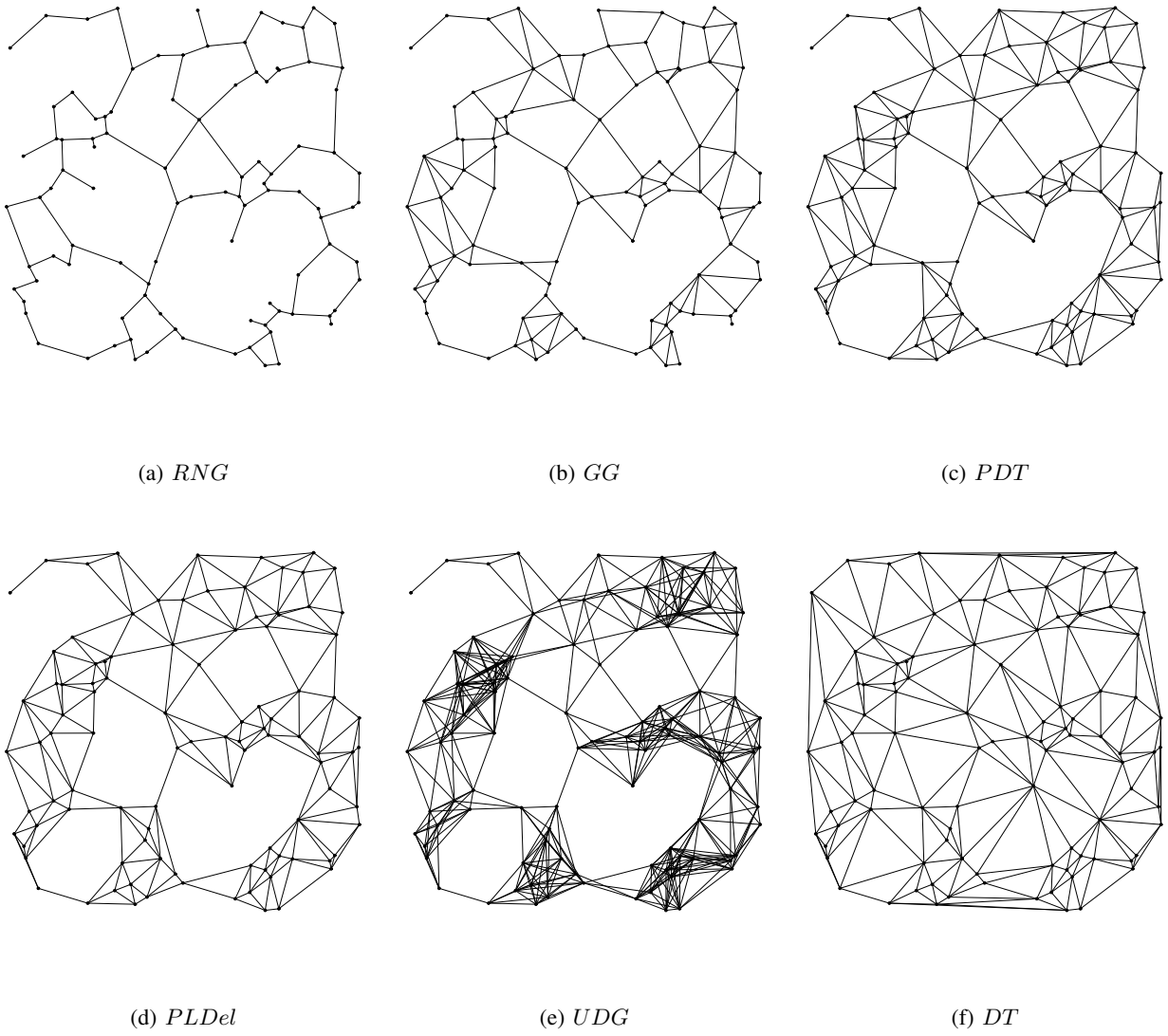
COMPARISON OF THE PRE-PROCESSING ALGORITHMS

Algorithm	Localized?	Neighborhood information	Cost	Steps	Graph model	$(4\sqrt{3}\pi)/9$-spanner
UDG	Yes	1 hop	$O(n \log n)$	0	UDG	Yes
RNG	Yes	1 hop	$O(n \log n)$	0	GM	No
GG	Yes	1 hop	$O(n \log n)$	0	GM	No
PDT	Yes	1 hop	$O(n \log n)$	0	UDG	No
Li *et al.* [18]	Yes	2 hop	$O(n \log n)$	4	UDG	Yes
FLDT1	Yes	2 hop	$O(n \log n)$	1	RNM	Yes
FLDT2	Yes	2 hop	$O(n \log n)$	1	UDG	Yes

we know for sure that FLDT1 still works as long as the RNM holds. This means that in between the ideal UDG conditions and the limits of the RNM model, while all the other triangulations fail, FLDT1 works and will create denser graphs (with more edges) than the GG at approximately the same signaling cost.

VI. EVALUATION

In this section, we compare the signaling cost of FLDT1 and FLDT2 versus the algorithm of [17] and its optimized version [18]. Before we make a quantitative evaluation of the algorithms, we start by comparing their most important features in Table I. All these algorithms are localized and have a communication cost of $O(n \log n)$ (we assume that nodes send at least one BEACON message with their own information). Some of them do not need extra messages from neighbors to create the graph (1 hop neighborhood), while others need at least some information about invisible nodes (relayed by the direct neighbors - 2 hop neighborhood). The algorithms also differ in the communication steps they use (besides one BEACON message that we do not count, but, as we point out in Section VI-B, this message can make a difference in dynamic scenarios) and in the communication model that they need to create a planar graph (GM, RNM or UDG). Finally, only some of them are $(4\sqrt{3}\pi)/9$-spanners of UDG (in scenarios where the UDG model holds). Figure 9 illustrates the graphs for the same set of 100 nodes.



(a) RNG (b) GG (c) PDT

(d) $PLDel$ (e) UDG (f) DT

Fig. 9. Example of graphs

A. Comparison of the Signaling Costs of the Algorithms

To compare the signaling cost of the algorithms and also to compare their performance (see Appendix A) we have always used the UDG model (given that some algorithms only work with this model). Since node density has a crucial impact on both the signaling cost and the performance of routing algorithms, we have randomly and uniformly distributed a variable number of nodes (between 80 and 600) inside a square of fixed side (7.5 times the communication range).

We believe that the most significant criterion of comparison of the algorithms that create $PLDel(V)$ graphs is the signaling cost. Hence, we depict in Figure 10 the average number of neighbors announced

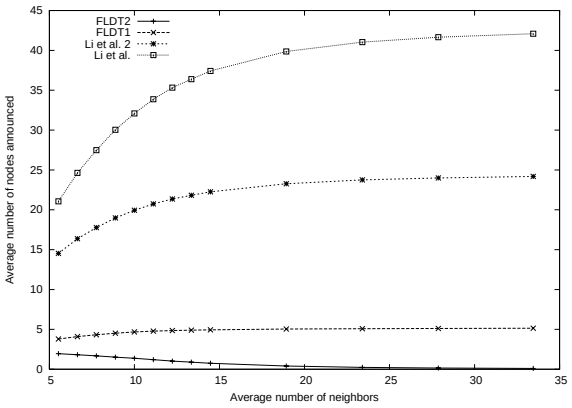


Fig. 10. Average number of nodes announced by each node

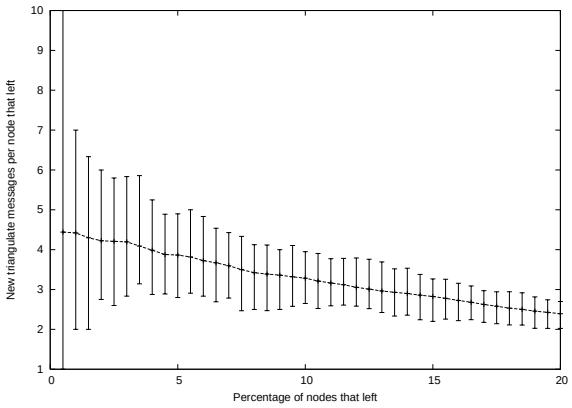


Fig. 11. Number of nodes that must resend their TRIANGULATE message per node that left

by each node, in the following algorithms: i) FLDT1, ii) FLDT2, iii) Li *et al.* [17] and Li *et al.* 2 [18]. To obtain the averages that we present in Figure 10, we created 400 different graphs for each one of the varying node densities. As we show in the Appendix A, the results of interest to us lie in the range around $5 - 20$ neighbors per node (for a node whose communication unit disk is entirely inside the simulation square). At lower densities, resulting topologies will most likely be disconnected, while at higher densities greedy outperforms any other right-hand algorithm. Unlike the corresponding graphic that we presented in [1], here we count the node that sends the message. This explains why the algorithm of Li *et al.* seems to announce more nodes here. On the other hand, in the algorithm of Li *et al.* 2, we use all the optimizations that the authors propose. This algorithm uses a little more than half the messages of the previous version to build a slightly different $PLDel(V)$ graph[5]. In our FLDT1 and FLDT2 algorithms, we used optimizations similar to the ones proposed by Li *et al.* 2 [18]. In particular, we only announce each node once in a TRIANGULATE message even if that node participates in two different triangles, thus saving one redundant announcement. This explains why we get results that are similar to the ones presented in [1], despite counting the sender of the message.

We can see that the number of nodes announced stabilizes in all the algorithms as the density increases.

[5]According to the wide definition of $PLDel(V)$ that we gave in Section III. Nevertheless, this difference seldom shows up.

For the densities of interest — up to 20 neighbors per node — FLDT1 can announce up to 4.6 and 7.9 fewer nodes than the algorithms Li *et al.* 2 and Li *et al.*, respectively. This result is even better for FLDT2 (58.2 and 99.7), which approaches 0 messages per node as the density of the network increases. Even in the lower densities, we never got more than an average of 2 nodes announced by node. Finally, we emphasize that even more important than the number of nodes announced is the number of messages used: while our algorithms need a single message, both algorithms of Li *et al.* need four. These results show that our algorithms build $PLDel$ very efficiently. While FLDT2 looks better from a theoretical perspective, we believe that for a practical use FLDT1 is, in fact, the best algorithm to create $PLDel(V)$ and is preferable than PDT, because it does not require nodes to be aware of their communication range. Additionally, FLDT1 is more robust, because it works in models that degenerate from UDG, like the RNM.

B. Dynamic Evaluation

In Figure 11, we depict the number of messages that are required to recreate the triangulation when a node leaves the network (we assumed that neighbors of the departing node eventually become aware of the fact and trigger the algorithm). We used 50 different graphs with 200 randomly and uniformly distributed nodes each and removed an increasing number of nodes. For each set of nodes that leaves, we recount which of the remaining nodes need to resend their triangulation. Although the average node degree is around 6, for a small churn rate, only 4.5 nodes that stay must resend their triangles for each node that leaves (the same would apply if the node was entering instead of leaving). This results from the fact that some of these nodes may have no difference in their TRIANGULATE message due to the rule of announcing only triangles when the local angle is greater or equal to $\pi/3$. The graphic shows a decreasing line, because as more and more nodes leave it becomes likelier that a single node sees two or more neighbors departing, thus contributing to a better average (at a cost of maintaining outdated information). Besides the average, the figure also shows the smallest and the largest number of nodes that had to send new TRIANGULATE messages in all the 50 graphs we tried.

In practice, nodes usually announce that they are still alive by sending a periodic beacon message: if a node fails to send that beacon for some time, its neighbors will consider that it left. Our algorithms are particularly well suited for this kind of setting, as TRIANGULATE messages can be easily piggybacked to (or even replace) BEACON messages. Therefore, when periodic BEACON messages are required, our algorithms can be implemented with no additional messages, becoming extremely competitive with regard to the Gabriel graph, the Relative Neighborhood graphs or PDT.

One interesting aspect of FLDT1 and FLDT2, that results from the use of a single message, is that even if links are lossy, e.g., due to collisions of packets, 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 [21]), triangulation in a stable setting will eventually be correct.

VII. Conclusions and Future Work

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 two algorithms, FLDT1 and FLDT2, to build a well-known graph called $PLDel(V)$. Our experimental results show that we can use $PLDel(V)$ either to substitute $UDG(V)$, when node density is small, or as a complementary graph that ensures routing convergence for all node densities.

FLDT1 and FLDT2 have a communication cost of $O(n \log n)$, which is within a constant of the optimal and require a single communication step, unlike previous algorithms that require 4 communication steps to create $PLDel(V)$. Among the algorithms that only work in the UDG model, FLDT2 is the best graph to build $PLDel(V)$ as it requires fewer messages than the remaining. On the contrary, in the more general Relative Neighborhood Model (RNM), which goes beyond the UDG model, FLDT1 is the only triangulation that works. Such graph model does not impose a precise circular communication range of ray 1 and allows nodes to have different ranges depending on the direction. Furthermore, in dynamic settings that require the exchange of beacon messages, our algorithms requires no more messages than

the algorithms used to build the very simple but inefficient GG or RNG or even the PDT. Therefore, due to their efficiency and due to the improved robustness of FLDT1, we believe that our algorithms have a practical relevance for position-based wireless *ad hoc* networks.

As we stated before, FLDT1 does not work in the GM. However, we believe that it is possible to introduce some changes in the algorithm to make it work under such model (e.g. by making all the nodes announcing all their triangles). The real challenge should be to make those changes and still ensure a communication cost of $O(n \log n)$ as well as a single communication step. We leave this problem as an open issue for future work.

REFERENCES

[1] Filipe Araújo and Luís Rodrigues. Fast localized delaunay triangulation. In *The 8th International Conference On Principles Of Distributed Systems (OPODIS 2004)*, pages 81–93, Grenoble, France, december 2004. Springer-Verlag, LNCS 3544.

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

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

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

[5] 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.

[6] D. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Computational Geometry*, July 1990.

[7] D. Eppstein. Spanning trees and spanners. In *Handbook of Computational Geometry*, pages 425–461. Elsevier North-Holland, Amsterdam, 2000.

[8] Greogry G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, Institute for Scientific Information, March 1987.

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

[10] 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.

[11] 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.

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

[13] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *22nd ACM Symposium on the Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 2003.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Xiang-Yang Li, Gruia Calinescu, Peng-Jun Wan, and Yu Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):1035–1047, October 2003.

[19] 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.

[20] 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.

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

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

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

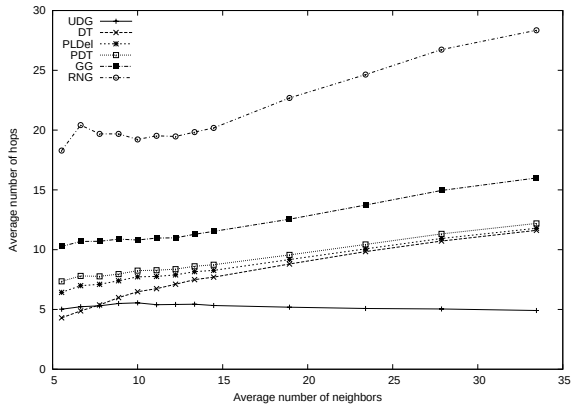


Fig. 12. Average number of hops per graph

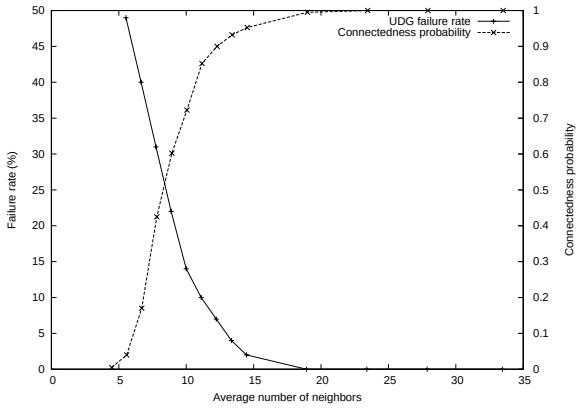


Fig. 13. Failure rate in the UDG

Appendix

A. Performance of the routing subgraphs

To complete the evaluation of our algorithms, we compare the routing performance in each of the following graphs: RNG, GG, PDT, $PLDel$, UDG and DT. We have used the GPSR routing algorithm [11] in all graphs, except in UDG, which is not planar. In UDG we have used the greedy routing algorithm. Results for the DT are depicted only to serve as a reference, because, as we have discussed before, such triangulation is not possible in a wireless environment. To depict the graphics of Figures 12 and 13, we have computed the averages of 20 random source-destination pairs in 400 different graphs for each different node density.

We show in Figure 12 the average path length in number of hops (for paths where greedy did not fail). Among the (possible) planar graphs, $PLDel$ achieves the best results, while the GG and the RNG achieve really bad results. An interesting conclusion to derive from this figure is that the DT benefits a lot from its long edges (longer than 1), especially for low densities. Figure 13 depicts the percentage of failures for the greedy routing algorithm in the UDG, versus the probability of having a connected topology, in the scenarios we evaluated. Both curves are functions of the average number of neighbors of a node, i.e., of density. Our results support the conclusion 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 (only slightly) shorter. More precisely, Figure 13 shows that the algorithms that we are comparing are most effective between 5 and 20 neighbors per node. To conclude, results of these figures suggest that we should use the greedy routing algorithm in UDG whenever possible, for performance reasons, and switch back to a right-hand rule algorithm and to a triangulation, like $PLDel$, whenever greedy fails.