2012

EFFICIENT GREEDY-FACE-GREEDY GEOGRAPHIC ROUTING PROTOCOLS IN MOBILE AD HOC AND SENSOR NETWORKS

Yan Sun
University of Kentucky, ysun2@uky.edu

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Yan Sun, Student

Dr. Mukesh Singhal, Major Professor

Dr. Raphael Finkel, Director of Graduate Studies

EFFICIENT GREEDY-FACE-GREEDY GEOGRAPHIC ROUTING PROTOCOLS IN MOBILE AD HOC AND SENSOR NETWORKS

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Yan Sun
Lexington, Kentucky
Director: Dr. Mukesh Singhal, Professor and Gartner Group Endowed
Chair in Networking
Lexington, Kentucky
2012

ABSTRACT OF DISSERTATION

EFFICIENT GREEDY-FACE-GREEDY GEOGRAPHIC ROUTING PROTOCOLS IN MOBILE AD HOC AND SENSOR NETWORKS

This thesis describes and develops two planarization algorithms for geographic routing and a geographic routing protocol for mobile ad hoc and sensor networks. As all nodes are mobile and there is no fixed infrastructure, the design of routing protocols is one of the most challenging issues in mobile ad hoc and sensor networks. In recent years, greedy-face-greedy (GFG) geographic routing protocols have been widely used, which need nodes to construct planar graphs as the underlying graphs for face routing.

Two kinds of planarization algorithms have been developed, idealized and realistic planarization algorithms, respectively. The idealized planarization algorithms make the ideal assumption that the original network graph is a *unit-disk graph (UDG)*. On the other hand, the realistic planarization algorithms do not need the original network to be a *UDG*.

We propose an idealized planarization algorithm, which constructs an *Edge Constrained Localized Delaunay graph (ECLDel)*. Compared to the existing *planarized localized Delaunay graph* [42], the construction of an *ECLDel* graph is far simpler, which reduces the communication cost and saves the network bandwidth.

We propose a *Pre-Processed Cross Link Detection Protocol (PPCLDP)*, which generates a planar spanning subgraph of the original network graph in realistic environments with obstacles. The proposed *PPCLDP* outperforms the existing *Cross Link Detection Protocol* [32] with much lower communication cost and better convergence time.

In GFG routing protocols, greedy routing may fail at concave nodes, in which case, face routing is applied to recover from the greedy routing failure. This may cause extra hops in routing in networks containing voids. We propose a *Hill-Area-Restricted (HAR)* routing protocol, which avoids the extra hops taken in the original GFG routing. Compared to the existing *Node Elevation Ad hoc Routing* [4], the proposed *HAR* guarantees the packet delivery and decreases the communication cost greatly.

KEYWORDS: Geographic routing, greedy routing, face routing, unit-disk graph, concave nodes.

Yan Sun
Jan. 08, 2012

EFFICIENT GREEDY-FACE-GREEDY GEOGRAPHIC ROUTING PROTOCOLS IN MOBILE AD HOC AND SENSOR NETWORKS

By
Yan Sun

Director of Dissertation
Dr. Mukesh Singhal

Director of Graduate Studies
Dr. Raphael Finkel

ACKNOWLEDGMENTS

I am thankful for the help of many people, without whom I could not have completed this dissertation.

I would like to express my gratitude to Dr. Mukesh Singhal, my Ph.D advisor and the chairman of my Ph.D committee. I have been lucky to work under his guidance. His encouragement, kindness and patience helped me define the research area I was really interested in. He always gives me valuable advice and inspiring comments. Discussions with him are real pleasure. Without his help, this dissertation would not have been possible.

I am also thankful to committee members Dr. D. Manivannan, Dr. Zongming Fei and Dr. Cai-Cheng Lu for their valuable comments.

Finally, I thank my husband, Qiangfeng Jiang for his encouragement and support. I have learned a lot from our discussions and work together.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Mobile Ad Hoc and Sensor Networks

This thesis develops two planarization algorithms for geographic routing and a geographic routing protocol for mobile ad hoc and sensor networks. In recent years, with the emergence of wireless devices such as PDAs and sensors, which need wireless communication, mobile ad hoc (MANET) and wireless sensor networks (WSN) have attracted a lot of attention. Mobile ad hoc and sensor networks are infrastructureless mobile networks, which contain wireless and mobile nodes that are connected in an arbitrary manner, without any infrastructure.

Because there are no fixed routers in mobile ad hoc and sensor networks, each node acts as both an end system and a router. A node can communicate directly only with nodes within its transmission range. When two nodes are not within each other's transmission range, the communication between them needs multi-hop routing, which needs the help of other mobile nodes to route packets between them. As all nodes are mobile and there is no fixed infrastructure, the design of routing protocols has become one of the most challenging issues in mobile ad hoc and sensor networks.

In mobile ad hoc and sensor networks, nodes usually have limited resources, i.e., memory and power, which requires routing protocols to be efficient with low overhead and low bandwidth consumption. This is the basis of the research described in this dissertation.

1.2 Routing Protocols in Mobile Ad Hoc and Sensor Networks

Routing protocols in mobile ad hoc and sensor networks can be divided to two groups: topology-based and position-based (geographic).

1.2.1 Topology-Based Routing Protocols in MANET

Topology-based routing protocols use the information of existing links in the network to route packets. A variety of topology-based routing protocols have been developed, which can be categorized as either *table-driven* or *source-initiated on-demand* routing protocols.

In *table-driven* routing protocols, each node maintains consistent and up-to-date routing information to every other node in the network, which requires periodic message flooding in the network. Each node needs to maintain one or more routing tables to store the routing information. Examples of *table-driven* routing protocols are DSDV [49] and WRP [46].

In *source-initiated on-demand* routing, routes are created only when they are required by source nodes. When a source node requires a route to a destination, it initiates a route discovery by message flooding in the network. Examples of *source-initiated on-demand* routing protocols are AODV [48] and DSR [27].

In topology-based routing protocols, message flooding in the network is typically needed for a node to get the routing information to other nodes. This makes topology-based routing protocols less scalable and less desirable for mobile ad hoc and sensor networks. Several articles include a survey of topology-based routing protocols [1, 2, 11, 51, 54].

1.2.2 Position-Based (Geographic) Routing Protocols in MANET

Position-based routing protocols use the information of the geographic position of nodes in the network to perform packet forwarding. The Global Position System (GPS) and some other positioning services [12, 23] help a mobile node know its own location. If nodes broadcast their locations locally (e.g., in Hello messages), each node will know the location

of all its neighbors.

In geographic routing, each mobile node does not need to maintain routing information achieved by message flooding. Instead, a node only needs to maintain the location of its neighbors, which is sufficient for it to select the next hop node for a packet. The low overhead of geographic routing makes it scalable and attractive for nodes with limited memory and power in mobile ad hoc and sensor networks. For example, in the recently proposed data-centric storage [38, 39, 52, 55, 57] for sensor networks, geographic routing protocols like GPSR [30] are used as the underlying routing protocols. Geographic routing protocols can be divided into heuristic and delivery-guaranteed routing protocols.

Examples of position-based (geographic) routing protocols are Compass [34], MFR [61], GEDIR [59], Face-2 [10], GPSR [30], AFR [36], and GOAFR$^+$ [35]. Several articles include a survey of geographic routing protocols [20, 45, 58].

Heuristic Geographic Routing Protocols

In heuristic geographic routing protocols, each node forwards a packet for a destination to a next hop node based on some heuristics. Typical heuristic geographic routing protocols [16, 34, 59, 61] use different heuristics, illustrated in Figure 1.1. Node s is a source node or a forwarding node of a packet with node d as the destination. The radius of the circle centered at node s is the transmission range of node s, so any node in the circle is a neighbor of s, like node p, q, m, g, and c in the figure. m' is the projection of node m on the dotted line sd.

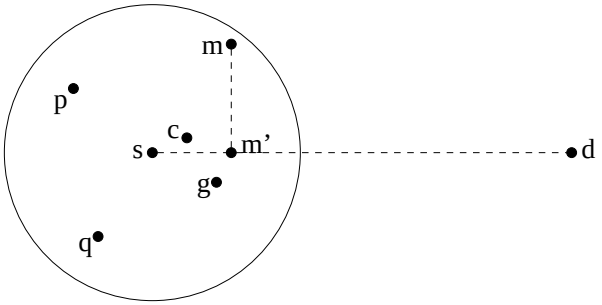


Figure 1.1: An illustration of heuristic geographic routing protocols

1. **Most Forward within Radius (MFR)**: Takagi et al. proposed the Most Forward within Radius (MFR) routing protocol [61], which is considered to be the first geographic routing protocol. In MFR, a source or a forwarding node s forwards a packet, with node d as the destination, to one of its neighbors that makes the most progress in the direction of d. That is, s forwards the packet to one of its neighbors whose projection on the line sd is closest to d. In Figure 1.1, node s will forward the packet to node m, because the projection of m on line sd, m', is closest to d than the projection of any other neighbors.

2. **Greedy Routing**: Finn proposed a greedy routing approach [16], in which node s forwards a packet to one of its neighbors that is closer to destination node d than s and any other neighbors of s, and in Figure 1.1, it is node g. It is easy to see that this kind of next hop node may not exist all the time, because node s may be closer to node d than all its neighbors. This is known as the local minimum phenomenon [69].

3. **Geographic Distance Routing (GEDIR)**: In GEDIR [59], node s forwards a packet to one of its neighbors that is closer to destination node d than any other neighbors of s, not necessarily closer to d than node s itself.

4. **Compass Routing**: In compass routing [34], node s forwards a packet destined to node d to one of its neighbors n such that the angle between sn and sd is the smallest. In Figure 1.1, node c is the next hop node selected by node s in this strategy.

The above heuristic routing protocols may not converge to find a path even though a path may exist, especially in sparse networks. Delivery-guaranteed geographic routing protocols can always find a path if one exists.

Delivery-Guaranteed (Greedy-Face-Greedy) Geographic Routing Protocols

Most delivery-guaranteed geographic routing protocols combine greedy and face routing and are called Greedy-Face-Greedy (GFG) routing protocols [58]. In GFG routing proto-

cols, greedy routing is used first, and when greedy routing fails, which means a node can not find a neighbor that is closer to the destination than itself and all its other neighbors, face routing is used to recover from this failure. Greedy routing will be used again when it is possible. Examples of GFG geographic routing protocols are GPSR [30], AFR [36], and GOAFR$^+$ [35]. GPSR [30] is shown in detail below.

Greedy Perimeter Stateless Routing (GPSR) Algorithm:

The Greedy Perimeter Stateless Routing (GPSR) [30] algorithm is a well-cited Greedy-Face-Greedy (GFG) routing protocol. Let s, d and p denote the source node, destination node and the packet, respectively. The GPSR protocol works as follows:

1. At the source node s, it first sets the routing mode of p to greedy and tries to route p by greedy routing, the description of which is in Algorithm 1.

   If the greedy routing fails, it sets the routing mode of p to perimeter and sets the node where p entered perimeter mode, denoted by e, to itself. Node e is for the later use to decide whether packet p can be returned to greedy mode. It then routes p by perimeter routing, the details of which are given in Algorithm 2.

2. When receiving packet p, a node, say u, checks if it is the destination node of p. If it is, it returns from the algorithm. Otherwise, it forwards p based on the routing mode of p as follows:

   (i) If the routing mode of p is greedy, it forwards p by greedy routing.

   (ii) If the routing mode of p is perimeter, it checks if p can be returned to greedy mode by checking if it is closer to d than node e, the node where p entered perimeter mode. If so, it changes the routing mode to greedy and forwards p by greedy routing. Otherwise, it forwards p by perimeter routing.

Details of the greedy routing algorithm, *Greedy-Routing(u, d, p)*, and of the perimeter (face) routing algorithm, *Face-Routing(u, d, p)*, are introduced in Algorithm 1 and Algo-

rithm 2, respectively, where u, d and p are the current node, the destination node and the packet that needs to be routed, respectively.

1. u checks if there is a neighbor node, say n, of it, which is closer to d than itself and any other neighbors of it.
2. If yes, u forwards p to node n.
3. Otherwise, the greedy routing fails and returns -1.

Algorithm 1: *The algorithm Greedy-Routing(u, d, p)*

1. If u is the source node, it forwards p to the first edge counterclockwise about u from the line ud.
2. If u is an intermediate forwarding node of p, and let v denote the previous hop node of p, it forwards p to the first edge counterclockwise about u from the line uv that does not intersect with the line ud.

Algorithm 2: *The algorithm Face-Routing(u, d, p)*

Perimeter (Face) routing in GFG routing protocols, like GPSR, makes each packet traverse along the faces, which intersect with the line segment from the source to the destination untill it reaches the destination or greedy routing can be returned. The traversal of packet is based on the right-hand rule [8], which must be applied on a planar graph without crossing edges to guarantee its correctness. Therefore, nodes in the network need to construct a planar spanning subgraph of the original network graph for face routing. In the next section, we introduce existing planarization algorithms and associated problems to be addressed.

1.3 Problems Addressed In the Dissertation

Two kinds of planarization algorithms have been developed for face routing recently, which we label idealized and realistic planarization algorithm, respectively. The idealized pla-

narization algorithms make an ideal assumption that the original network graph is a *unit-disk graph (UDG)*, which means there is an edge incident on two nodes if and only if the Euclidean distance between them is no more than the transmission range[1]. On the contrary, the realistic planarization algorithms do not need the original network to be a UDG. These algorithms have a planar spanning subgraph of the original network graph under realistic environments.

1.3.1 Idealized Planarization Algorithms for Face Routing

Idealized planarization algorithms make an ideal assumption that the original network graph is a *Unit-Disk Graph (UDG)*. Under this assumption, a planarization algorithm to construct the *Gabriel graph (GG)* [18], which is a planar spanning subgraph of the original *UDG*, is commonly used for face routing.

Let *UDG(N, E)* denote the *unit-disk graph* of the ad hoc or sensor network, where N is the set of all nodes in the network and E is the set of all edges in the graph. Given two nodes a and b, the *Gabriel graph (GG)* contains an edge ab if $ab \in E$ and if the interior of the circle with ab as diameter does not contain any other node (known as a witness) in N.

Another idealized planarization algorithm to construct the *Relative Neighborhood Graph (RNG)* [62], which is a planar spanning subgraph of the original *UDG*, is also commonly used for face routing. It contains an edge ab if $ab \in E$ and there is no node (known as a witness) $c \in N$ such that $|ac| < |ab|$ and $|bc| < |ab|$.

Both of the above idealized planarization algorithms are commonly used, because they can be constructed distributively and easily by each node. For instance, *GG* is used by Bose et al. [10] and *RNG* is used by Karp et al. [30]. However, *GG* and *RNG* are relatively sparse, which results in long routes for geographic routing on them.

Li et al. [42] proposed an idealized planarization algorithm to construct a *planarized localized Delaunay graph, PLDel*. *PLDel* graph is a planar t-spanner of *UDG(N, E)*, which

[1] Assuming all mobile nodes in the network have the same transmission range.

is denser than *GG* and *RNG*. The algorithm to construct a *PLDel(N)* contains two sections.

In the first section, a *1-localized Delaunay graph*, $LDel^{(1)}(N)$, is constructed. It contains all *Gabriel* edges and all *1-localized Delaunay triangle*, the circumcircle of which does not contain any 1-hop neighbors of any of its three vertices. The $LDel^{(1)}(N)$ is a t-spanner of *UDG(N, E)*, but it may not be planar.

In the second section, by removing the intersections in $LDel^{(1)}(N)$, the graph *PLDel(N)* is constructed, which is a planar t-spanner of the original *UDG*.

The algorithm to construct the *PLDel* needs 1-hop neighborhood information. However, the construction of *PLDel* is very complex and not efficient enough because each node needs to broadcast several rounds of messages, which results in high communication cost and makes their algorithm converge slowly.

We propose an idealized planarization algorithm to construct an *Edge Constrained Localized Delaunay graph*, denoted by *ECLDel*, as the underlying graph for face routing. We prove that the *ECLDel* is a planar t-spanner of the *unit-disk* graph, which is denser than *GG* and *RNG*. Geographic routing on *ECLDel* is as efficient as on the previous work of *PLDel* in terms of path length (hop count). However, the construction of an *ECLDel* graph is far more simple and it converges faster. In addition, both the number and the size of messages broadcast by each node in the construction of *ECLDel* graph are significantly decreased, which reduces communication cost and saves the network bandwidth and node power.

1.3.2 Realistic Planarization Algorithms for Face Routing

In realistic environments, the assumption that the original network is a *UDG* may be violated in the following three situations:

1. Obstacles may exist in two neighboring nodes, and an edge may not exist between them (i.e., they may not be able to communicate with each other directly) even though the *Euclidean* distance between them is less than the transmission range. This thesis focuses on this particular case.

2. Nodes may have different transmission ranges, making links between nodes unidirectional.

3. The location information obtained by *GPS* or other systems may be inaccurate and have some error. Seada et al. [56] describe that many state-of-the-art techniques usually cause 10% (of the transmission range) or more in location error. This will result in an error in calculating the *Euclidean* distance between two nodes.

The violation of the assumption of *UDG*, in which an edge may not exist between two nodes (even though the *Euclidean* distance between them is less than the transmission range), causes the idealized *UDG* planarization algorithms not to work correctly. As a result, realistic planarization algorithms must be developed.

The *Cross Link Detection Protocol (CLDP)* [32], to our knowledge, is the only one that uses a realistic planarization algorithm, which makes face routing in *GFG* geographic routing protocols, like *GPSR*, work correctly under realistic conditions with obstacles.

The *CLDP* produces an almost planar spanning subgraph of the original realistic network graph. In *CLDP*, a node needs to probe each link attached to it in the original network graph to detect pairs of links that intersect. If a node detects such pairs of cross-links, it may need further probing of a link to decide if the link should be kept or removed.

The probing of a link in *CLDP* requires traversal of the network graph using the *Right-Hand Rule (RHR)*, which in some cases may cause a long face with a large number of edges being traversed. This results in large communication cost because each traversal of an edge needs a message broadcast. In addition, for some of its links, a node may need several rounds of probing (several rounds of traversal of faces) to make a decision, which makes *CLDP* converge slowly.

We propose a *Pre-Processed Cross Link Detection Protocol (PPCLDP)* containing a realistic planarization algorithm, which generates an almost planar spanning subgraph of the original network graph in realistic environments with obstacles. The proposed *PPCLDP*

improves the existing *CLDP* by adding a *2-hop Cross Link Pre-Processing (CLPP)* algorithm. In the *CLPP* algorithm, a node can detect *2-hop cross links* of any links attached to it and can decide whether to keep or remove the links by exchanging a few messages with its neighbors. This way, a node does not need to probe these links using the *Right-Hand Rule (RHR)*, which may cause traversal of long faces, and does not need several rounds of probing for these. This decreases the total number of messages broadcast by nodes significantly, which makes *PPCLDP* outperform *CLDP* with a much lower communication cost and better convergence time.

1.3.3 GFG Geographic Routing Protocols for Networks with Voids

In most existing GFG geographic routing protocols, like *GPSR* [30], greedy routing is applied first, which may fail at a concave node, which is a node closer to the destination than any of its neighbors [30]. The reason for this is that in geographic routing protocols, only local information of each neighbor's position is known to each node, hence, a node can not make routing decisions based on the whole network topology. Therefore, nodes may not select next hop neighbors wisely in some networks with special topologies, e.g., networks with voids or obstacles. This may cause packets to enter concave areas and reach concave nodes, and cause greedy routing fail. Face routing is applied to recover from greedy routing failures, which may cause many extra hops in routing and decrease the routing efficiency.

Noa et al. [4] propose a *Node Elevation Ad hoc Routing (NEAR)* protocol to improve the routing efficiency. It consists of three algorithms, which are a node reposition algorithm, a void bypass algorithm and a routing algorithm. It improves the overall efficiency of greedy and perimeter routing of the original GFG routing. However, there are two problems with *NEAR*.

First, the Right-Hand Rule (RHR) is used on a non-planar graph in the void bypass algorithm, which fails to find paths around voids in some cases. As a result, the packet delivery is not guaranteed in the network, which is not desirable in mobile ad hoc and sensor

networks. Second, the void bypass algorithm in *NEAR* incurs tremendous communication cost in transferring control messages.

We propose a *Hill-Area-Restricted (HAR)* geographic routing protocol, which avoids the extra hops required in the original GFG routing, making it more efficient in hop count. Compared to the previous work of *a Node Elevation Ad Hoc Routing (NEAR)* [4], the proposed HAR guarantees the packet delivery and greatly decreases the communication cost. This makes the HAR more desirable for mobile ad hoc and sensor networks.

1.4 Organization of the Dissertation

In Chapter 2, we present an idealized planarization algorithm, which constructs an *Edge Constrained Localized Delaunay graph (ECLDel)*, as the underlying graph for face routing. We prove that the *ECLDel* is a planar t-spanner of the *unit-disk* graph. In Chapter 3, we present our *Pre-Processed Cross Link Detection Protocol (PPCLDP)*. It uses a realistic planarization algorithm, which generates an almost planar spanning subgraph of the original network graph in realistic environments with obstacles. We introduce the *Hill-Area-Restricted (HAR)* geographic routing protocol in Chapter 4. The *HAR* protocol avoids the extra hops taken in the original GFG routing in networks containing voids, which makes it more efficient with respect to the hop count. Finally, we conclude the dissertation and describe our future work in Chapter 5.

Chapter 2

An Edge Constrained Localized Delaunay Graph for Geographic Routing in MANET

2.1 Introduction

In recent years, a variety of position-based (geographic) routing protocols have been developed for mobile ad hoc and sensor networks, such as Compass [34], MFR [61], GEDIR [59], Face-2 [10], GPSR [30], AFR [36] and GOAFR$^+$ [35].

Most geographic routing protocols combine greedy and face routing, which are called Greedy-Face-Greedy (GFG) routing protocols [58]. In GFG routing protocols, greedy routing is used first, and when this fails, i.e., when a node can not find a neighbor that is closer to the destination than itself and all its other neighbors, face routing is used to recover from this failure. Greedy routing will be used again when it is possible. Examples of GFG geographic routing protocols are GPSR [30], AFR [36], and GOAFR$^+$ [35].

Face routing is based on the right-hand rule [8] and must be applied on a planar graph without crossing edges to guarantee its correctness. In a planar graph that is composed of faces, face routing makes each packet traverse along the faces, which intersect with the line segment from the source to the destination, until the destination is reached or greedy routing can be returned. This requires nodes in the network to construct a planar spanning subgraph of the original network graph.

Many existing planarization algorithms, which we label idealized planarization algorithms, make the idealized assumption that the original network graph is a *unit-disk graph (UDG)*, where there is an edge incident on two nodes if and only if the Euclidean distance between them is no more than the transmission range[1].

Two idealized planarization algorithms commonly used are the *Gabriel graph (GG)* [18] and the *relative neighborhood graph (RNG)* [62], because they can be constructed distributively and easily by each node. For instance, *GG* is used by Bose et al. [10] and *RNG* is used by Karp et al. [30]. However, *GG* and *RNG* are relatively sparse, which results in long routes for geographic routing on them.

To improve the efficiency of geographic routing, i.e., shortening the routes, some denser graphs, e.g., planar t-spanners[2] of *UDG* [19, 42, 65], are proposed as the underlying graphs.

Li et al. [42] propose a *planarized localized Delaunay graph, PLDel*, which is a planar t-spanner of *UDG*, as the underlying graph for geographic routing. Their algorithm to construct the *PLDel* is very complex and converges slowly, because each node needs to broadcast several messages, which results in a high communication cost.

We propose an *Edge Constrained Localized Delaunay graph, ECLDel*, as the underlying graph for geographic routing. In this chapter:

1. We prove that the *ECLDel* is a planar t-spanner of the original *unit-disk* graph.

2. We develop an algorithm to construct the *ECLDel* graph, which can be run by each node distributively with 1-hop neighborhood information.

3. Compared to the previous work of constructing a *planarized localized Delaunay graph, PLDel*, our algorithm to construct the *ECLDel* is much simpler and converges faster. This is because:

[1] Suppose all mobile nodes in the network have the same transmission range.

[2] A graph G' is a t-spanner of a graph G, if G' is a spanning subgraph of G and the shortest path length between any two nodes in G' is at most t times the shortest path length between the two nodes in G. t is a positive real constant and is called the $length\ stretch\ factor$.

(i) We significantly reduce the number of messages broadcast by each node from five rounds (each round may contain several messages) to only two messages; and

(ii) We define two new types of edges, the *Intersecting Gabriel (IG) edges* and the *Unaware Intersection (UI) edges*, which are constrained in the *ECLDel* graph. These edges help significantly reduce the size of messages broadcast by each node. The decrease in both the number and the size of messages broadcast by each node reduces the communication cost, and saves the network both bandwidth and node power, which is desirable in mobile ad hoc and sensor networks.

4. Our simulation shows that the communication cost decided by the average number of messages and the average size of messages (the number of neighbor nodes in messages) broadcast by each node is, respectively, 65% and 42% less in the construction of *ECLDel* than in *PLDel*.

Section 2.2 below presents the rationale behind our research. Section 2.3 introduces preliminaries. Section 2.4 presents the *Edge Constrained Localized Delaunay graph, ECLDel*, and proves it is a planar t-spanner of the *unit-disk graph*. Section 2.5 describes an algorithm to construct an *ECLDel* graph. Section 2.6 presents the simulation results on the performance of geographic routing on *ECLDel* and other underlying graphs, and on the cost of constructing *PLDel* and *ECLDel*. Section 2.7 describes the related work and Section 2.8 concludes the chapter.

2.2 Research Rationale

2.2.1 Assumptions

1. Accurate positioning service: We assume that in mobile ad hoc and sensor networks, each node has GPS or other positioning services [12, 23] that can provide its accurate position of. If nodes broadcast their locations, each node will know the accurate

location of all its neighbors.

2. *Unit-disk graph:* We also assume that all mobile nodes in the network have the same transmission range. No obstacles exist between any two nodes that are in each other's transmission range, so they can communicate with each other directly. In this case, mobile nodes in the network construct a *unit-disk graph* (*UDG*), because there is an edge between two nodes if and only if the Euclidean distance between them is no more than the transmission range.

2.2.2 Research Rationale

Greedy-Face-Greedy (GFG) routing has been developed as a kind of optimal delivery-guaranteed geographic routing, and has become the main trend in this field. In GFG routing, face routing is used when greedy routing fails to recover from the failure. As discussed previously, face routing based on the right-hand rule [8] must be applied on a planar graph, in which no crossing edges exist, to guarantee its correctness. Therefore, planar spanning subgraphs of the original network graph, *UDG*, must be constructed for GFG routing.

The *Gabriel graph* (*GG*) [18] and *relative neighborhood graph* (*RNG*) [62] are commonly used as the underlying planar graphs for GFG routing. However, both *GG* and *RNG* are relatively sparse, which makes GFG routing inefficient with long routes. Recently, many researchers have focused on the construction of denser graphs, e.g., planar t-spanners of *UDG*, as the underlying graphs [5, 19, 42, 65], which makes GFG routing much more efficient with shorter routes.

Li et al. [42] propose a planar t-spanner of *UDG*, called a *planarized localized Delaunay graph*, *PLDel*, as the underlying graph for GFG geographic routing, which can be constructed by each node distributively. Their algorithm to construct the *PLDel* is very complex and is not efficient because each node needs to broadcast several messages, which makes their algorithm converge slowly. This is the rationale behind our study.

In this chapter, we develop a simpler and more efficient algorithm to construct a planar

t-spanner of *UDG* for GFG geographic routing. In our algorithm, we do not attempt to decrease the asymptotic communication cost, which is near optimal in the algorithm to construct a *PLDel*. However, we do significantly decrease the number of messages and the size of messages broadcast by each node in the construction of *ECLDel*. This results in a much lower communication cost and makes our algorithm converge faster, which are desirable features in mobile ad hoc and sensor networks.

2.3 Preliminaries

Before presenting *ECLDel*, some background information is needed. Let *UDG(N, E)* denote the *unit-disk graph* of the ad hoc or sensor network, where N is the set of all nodes in the network and E is the set of all edges in the graph. Given two nodes a and b, the Euclidean distance between them is denoted by $|ab|$.

2.3.1 The Gabriel Graph

The *Gabriel graph* [18][3], denoted by *GG*, is a planar spanning subgraph of *UDG(N, E)*. It contains an edge ab if $ab \in E$ and if the interior of the circle with ab as diameter does not contain any other node (known as a witness) in N. As shown in Figure 2.1, the shaded area is the circle with a diameter of ab, in which if there is no witness c, then ab will be kept by both a and b as an edge in the *Gabriel graph*.

2.3.2 Relative Neighborhood Graph

The *relative neighborhood graph* [62][4], denoted by *RNG*, is also a planar spanning subgraph of *UDG(N, E)*. It contains an edge ab if $ab \in E$ and there is no node (known as a witness) $c \in N$ such that $|ac| < |ab|$ and $|bc| < |ab|$. For example, in Figure 2.2, the shaded area is the intersection area of two circles, both of which have $|ab|$ as their radius and the

[3] In the rest of the chapter, all *Gabriel graphs* are extracted from the *UDG*.
[4] In the rest of the chapter, all *relative neighborhood graphs* are extracted from the *UDG*.

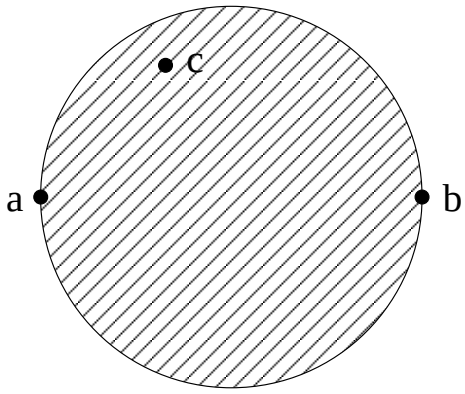


Figure 2.1: Deciding whether an edge is in the *GG graph*

centers of which are a and b, respectively. If there is no witness c in the shaded area, then edge ab will be kept by both a and b as an edge in the *RNG graph*.

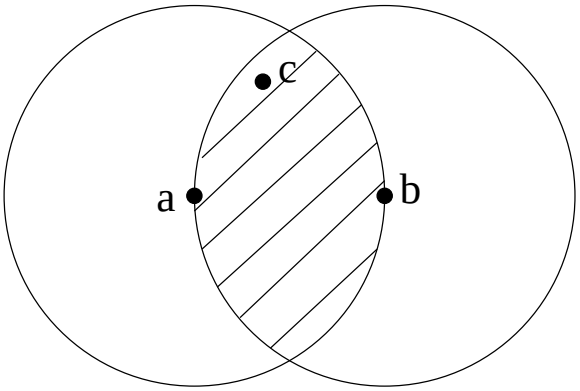


Figure 2.2: Deciding whether an edge is in the *RNG graph*

The length stretch factors of *GG* and *RNG* are $n - 1$ and $\sqrt{n-1}$, respectively [63], where n is the number of nodes in the network. So *GG* and *RNG* are not t-spanner of the *UDG(N, E)*.

2.3.3 Voronoi Diagram and Delaunay Triangulation

The *Voronoi region* of a node a is a region N in which each node is closer to node a than to any other node in N. The *Voronoi regions* of all nodes in N construct the *Voronoi diagram*. *Delaunay triangulation* is the dual of the *Voronoi diagram*. *Delaunay triangu-*

lation contains an edge ab if and only if the *Voronoi regions* of a and b share a common boundary [50]. The *Delaunay triangulation*, denoted by *Del(N)*, is a planar t-spanner of the completed Euclidean graph [14]. Each triangle in the *Delaunay triangulation* is called a *Delaunay triangle*, which has the important property that the interior of its circumcircle does not contain any other node in N.

Figure 2.3 below shows an example of the *Voronoi diagram* and *Delaunay triangulation*, where the dotted lines construct the *Voronoi diagram* and the solid lines construct the *Delaunay triangulation*.

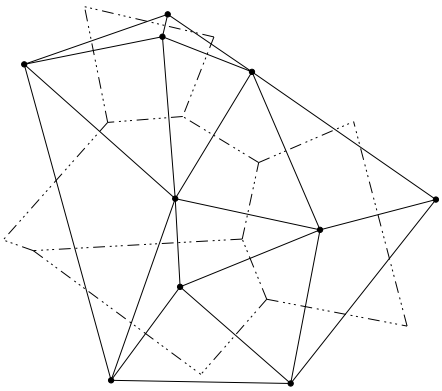


Figure 2.3: An example of *Voronoi diagram* and *Delaunay triangulation*

2.3.4 Planarized Localized Delaunay Graph (PLDel)

k-Hop Neighbors

Nodes within the transmission range of a node a are called 1-hop neighbors of a, which can be reached by a directly. Nodes that can be reached by node a *within* (less than or equal to) k hops are called k-hop neighbors of a.

Planarized Localized Delaunay Graph

Li et al. [42] proposed a *k-localized Delaunay graph*, denoted by *LDel*$^{(k)}$*(N)* $(k \geq 1)$, which contains all *Gabriel edges* and all *k-localized Delaunay triangles*[5].

[5] A triangle is a *k-localized Delaunay triangle* if the interior of its circumcircle does not contain any k-hop neighbors of any of its three vertices.

Li et al. [42] proved that *LDel*$^{(k)}$*(N)* $(k \geq 1)$ is a t-spanner of *UDG(N, E)*. They illustrated that *LDel*$^{(k)}$*(N)* is planar when $k \geq 2$, but its construction needs at least 2-hop neighborhood information from each node, which is undesirable. The construction of *LDel*$^{(1)}$*(N)* needs 1-hop neighborhood information, but it has been shown that it may not be planar.

Li et al. proposed the *planarized localized Delaunay graph*, denoted by *PLDel(N)*, which is a planar t-spanner of *UDG(N, E)* and can be constructed with 1-hop neighborhood information. The algorithm to construct a *PLDel(N)* contains two parts as follows:

Part I:

i. A node, say a, computes the *Delaunay triangulation*, *Del(N(a))*. $N(a)$ denotes node a's 1-hop neighbors, including itself.

ii. For a triangle from *Del(N(a))*, say $\triangle abc$, if all the three edges have length at most the transmission range, and angle $\angle bac \geq \pi/3$, node a **broadcasts** a message *Proposal(a, b, c)*.

iii. When receiving the message *Proposal(a, b, c)*, a node, say b, checks whether $\triangle abc$ belongs to *Del(N(b))*. If yes, b **broadcasts** a message, *Accept(a, b, c)*. Otherwise, it rejects the proposal by **broadcasting** message *Reject(a, b, c)*.

iv. Node a keeps edges ab and ac as the incident edges of it if $\triangle abc$ is in *Del(N(a))*, and both b and c have sent either *Accept(a, b, c)* or *Proposal(a, b, c)*.

The graph constructed by this part is the *1-localized Delaunay graph*, *LDel*$^{(1)}$*(N)*. Each triangle in the *LDel*$^{(1)}$*(N)* is called a *1-localized Delaunay triangle*, *LDel*$^{(1)}\triangle$. Part II removes the intersections in *LDel*$^{(1)}$*(N)*.

Part II:

i. A node a **broadcasts** the *Gabriel edges* incident on it and **broadcasts** the triangles of the *LDel*$^{(1)}$*(N)* incident on it.

ii. For two intersected triangles $\triangle abc$ and $\triangle def$ known by node a, node a removes the triangle $\triangle abc$ if its circumcircle contains one of the nodes d, e and f.

iii. Node a removes any triangle of the *LDel*$^{(1)}$*(N)* incident on it that intersects with any *Gabriel edge* it received from other nodes.

iv. Node a **broadcasts** all triangles incident on it which it has not removed in the previous steps.

v. Node a keeps the edge ab as the incident edges of it in *PLDel(N)* if it is a *Gabriel edge*, or if there is a triangle $\triangle abc$, such that a, b, and c have all announced they have not removed the triangle $\triangle abc$ in step ii and step iii.

The above algorithm constructs a *PLDel(N)* graph. The communication cost in the construction is $O(n \lg n)$, where n is the number of nodes in the network. It is easy to see that the construction of *PLDel(N)* is very complex and that each node needs to **broadcast** five rounds of messages (in addition, each round may contain several messages). This makes the algorithm inefficient and it converges slowly. In the following section, we present a much more efficient algorithm to construct a planar t-spanner of *UDG*.

2.4 Edge Constrained Localized Delaunay Graph

In this section, we define two new kinds of edges as the *Constrained edges*, which belong to the *UDG* and are constrained in the proposed *Edge Constrained Localized Delaunay graph*, *ECLDel*. We also prove that the *ECLDel* is a planar t-spanner of *UDG*.

2.4.1 Edges Constrained in ECLDel

We define two new kinds of edges, *Intersecting Gabriel (IG) edges* and *Unaware Intersection (UI) edges*, which are constrained in the *ECLDel*.

Before defining the edges, we introduce some notations. The transmission range of each mobile node is denoted by R. Given any three nodes p, q, and r, the triangle made by them is denoted by $\triangle pqr$. The circumcircle made by them is denoted by $\odot pqr$. $\odot pq$ denotes the circle with diameter pq, and $\odot p$ denotes the circle with p as the center and R as the radius. The angle ([0, π]) between edges pq and pr is denoted by $\angle qpr$ or $\angle rpq$.

Intersecting Gabriel (IG) edges

First, we define a new kind of edges, the *Intersecting Gabriel (IG) edges*.

Definition 1 *Any edge in* UDG *that intersects with a Gabriel edge is called an Intersecting Gabriel (IG) edge.*

Lemma 1 *If cd is an IG edge that intersects with a Gabriel edge ab, then at least one of a and b is a common neighbor of both c and d.*

Proof: Let cd intersect with ab at e. Let o be the center of $\odot ab$. Because ab is a *Gabriel edge*, both c and d are outside $\odot ab$. Let cd intersect with $\odot ab$ at c' and d', respectively. There are three cases based on the location of e.

Case 1: e and b are on the same side of o, as shown in Figure 2.4 below. Then $\angle cbd > \angle c'bd' > \pi/2$. This implies that $|bc| < |cd| \leq R$ and $|bd| < |cd| \leq R$. Therefore, b is a common neighbor of both c and d.

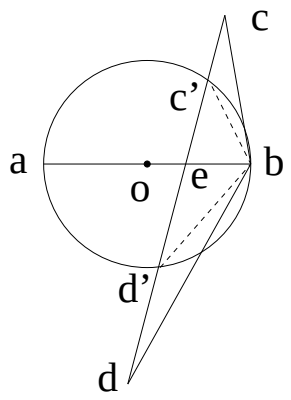


Figure 2.4: e and b are on the same side of o

Case 2: e and a are on the same side of o. Based on the reasoning in Case 1 above, it is easy to see that a is a common neighbor of both c and d.

Case 3: e is exactly on o. Also based on the reasoning in Case 1, it is easy to see that both a and b are common neighbors of c and d. Thus, the lemma follows. □

Corollary 1 *Since a node broadcasts all Gabriel edges incident on it, each node has the knowledge of all IG edges incident on it.*

This is accomplished by a node by checking whether an edge incident on it intersects with any *Gabriel edge* broadcast by its neighbors.

Unaware Intersection (UI) Edges

Next, we define the other type of edges, namely, the *Unaware Intersection (UI) edges*. Let $N(a)$ denote the set of 1-hop neighbors of node a.

Definition 2 *For a Non-Gabriel edge cd incident on c, if $\exists a \in N(c)$, $b \in N(c)$, which makes ab intersect with cd with $|ad| > R$ and $|bd| > R$, as shown in Figure 2.5, then ab is called an Unaware Intersection (UI) edge, because neither a nor b knows about the intersection. Node c is called a discoverer of the UI edge ab. cd is called a bridge edge for c to discover the UI edge ab.*

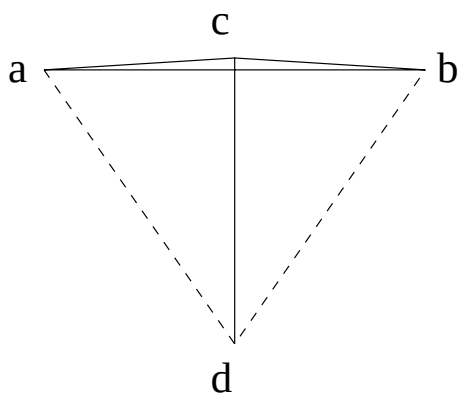


Figure 2.5: An *UI* edge (dotted lines are those with length more than R)

Because both a and b are neighbors of the discoverer c, if c broadcasts ab as an *UI* edge, then both a and b know that ab is an *UI* edge. In fact, a node discovers very few *UI* edges, as discussed next.

Corollary 2 *If a node, say c, broadcasts the UI edge, say ab, it discovers, then both a and b know that ab is an UI edge.*

The following Lemma 2, Corollary 3, and Lemma 3 show characterizations of UI edges, which helps explain why the number of UI edges each node discovers is so low. This results in a low communication cost for each node to broadcast the UI edges, which is required in our algorithm to construct the *ECLDel* introduced later.

Lemma 2 *If c is a discoverer of an UI edge ab with a bridge edge cd, then $|cd| > (\sqrt{3}/2)R$.*

Proof: The proof uses Figure 2.6 below. Let the area inside $\odot c$ be denoted by A_1, and the area inside $\odot d$ be denoted by A_2. Let A_3 denote the area that is in A_1 but not in A_2, which means $A_3 = A_1 - A_1 \cap A_2$, shown as the shaded area in Figure 2.6.

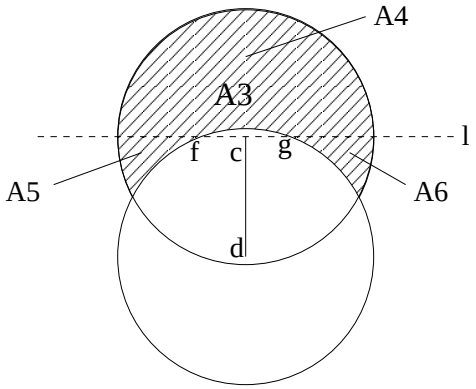


Figure 2.6: Property of an *UI* edge

Based on Definition 2, we know $|ac| \leq R$, $|bc| \leq R$, $|ad| > R$, and $|bd| > R$, which means both a and b are in area A_3.

Let dotted line l incident on c be perpendicular to cd. It intersects with $\odot d$ at f and g. Both a and b are in A_3 and intersect with cd, so $|fg| < |ab| \leq R$. This implies that $|cg| = |fg|/2 < R/2$. Since $|dg| = R$, then $|cd| > (\sqrt{3}/2)R$. □

Corollary 3 *If c is a discoverer of an UI edge ab with a bridge edge cd, and as in Figure 2.6, line l separates the area A_3 into three subareas, A_4, A_5, and A_6, which means $A_3 = A_4 \cup A_5 \cup A_6$, then at least one of a and b is in the area $A_5 \cup A_6$.*

Proof: ab intersects with cd, so a and b can not be both in A_4. Since both a and b are in A_3, and $A_3 = A_4 \cup A_5 \cup A_6$, the corollary follows. □

Lemma 3 *If c is a discoverer of an UI edge ab with a bridge edge cd, then $\angle acb > 2\pi/3$.*

Proof: Based on Definition 2, we know that $|bc| \leq R$, $|bd| > R$ and $|cd| \leq R$, as shown in Figure 2.5, so $|bd| > |bc|$ and $|bd| > |cd|$. This implies that $\angle bcd > \pi/3$. For the same reason, $\angle acd > \pi/3$. Therefore, $\angle acb = \angle acd + \angle bcd > 2\pi/3$. □

Lemma 2, Corollary 3, and Lemma 3 imply that the number of *UI* edges each node can discover is very low. Our simulations show that the *UI* edges each node discovers is 0.12 on the average, which generates a very low communication cost for broadcasting *UI* edges.

Lemma 4 *If an edge ab is an UI edge, it is not a Gabriel edge.*

Proof: Let c be a discoverer of an *UI* edge ab by a bridge edge cd. Then based on Definition 2, ab intersects with cd and neither a nor b is the neighbor of d.

Suppose ab is a *Gabriel edge* and cd is an *Intersecting Gabriel (IG) edge*. Based on Lemma 1, at least one of a and b is the common neighbor of both c and d. This contradicts that neither a nor b is the neighbor of d. Hence, the lemma follows. □

Constrained edges

With the definitions of *Intersecting Gabriel (IG) edges* and *Unaware Intersection (UI) edges*, we are ready to introduce *Constrained edges*.

Definition 3 *If an edge is an Intersecting Gabriel (IG) edge or an Unaware Intersection (UI) edge (sometimes an IG edge may also be an UI edge), then it is called a Constrained edge.*

In the rest of this chapter, the set of all *Constrained edges* that are constrained in the *ECLDel* is denoted by *CE*.

2.4.2 Edge Constrained Localized Delaunay Graph

With the knowledge of *Constrained edges*, in this section, we define the *Edge Constrained Localized Delaunay graph*, denoted by *ECLDel(N)*.

Definition 4 *For any three nodes a, b, and c of N, if the interior of $\odot abc$ does not contain any 1-hop neighbor of a, b, or c, and each edge of $\triangle abc$ has a length of no more than R and is not a Constrained edge (which means each edge of $\triangle abc$ belongs to $E - CE$), then $\triangle abc$ is called an Edge Constrained Localized Delaunay triangle, denoted by ECLDel$\triangle$.*

Definition 5 *The Edge Constrained Localized Delaunay graph, denoted by ECLDel(N), contains all Gabriel edges and all edges of ECLDel$\triangle$s.*

2.4.3 ECLDel is a t-Spanner

In this section, we prove that the *ECLDel* is a t-spanner of the *UDG*. Recall that the *2-localized Delaunay graph*, *LDel$^{(2)}$(N)*, is a planar t-spanner of *UDG(N, E)* [42]. It contains all *Gabriel edges* and all *2-localized Delaunay triangles*[6]. By proving that it is a subgraph of *ECLDel(N)*, we prove that *ECLDel(N)* is a t-spanner of *UDG(N, E)*.

Lemma 5 *None of the edges of a 2-localized Delaunay triangle is an Intersecting Gabriel (IG) edge.*

Proof: LDel$^{(2)}$(N) contains all *Gabriel edges* and is planar, so it does not contain an *IG* edge. The lemma follows. □

Lemma 6 *None of the edges of a 2-localized Delaunay triangle is an Unaware Intersection (UI) edge.*

Proof: Suppose there is a *2-localized Delaunay triangle* $\triangle abc$ with an *UI* edge ab. Assume that node e is a discoverer of *UI* edge ab through a bridge edge ed. Based on

[6] A *2-localized Delaunay triangle*, say $\triangle abc$, satisfies the requirement that the interior of $\odot abc$ does not contain any 2-hop neighbor of a, b, or c, and each edge of $\triangle abc$ has length no more than R, (which means each edge of $\triangle abc$ belongs to E).

Definition 2, both a and b are 1-hop neighbors of e, and ab intersects with edge ed with $|ad| > R$ and $|bd| > R$. This implies d is a 2-hop neighbor of a and b. Note that e is a 1-hop neighbor and also a 2-hop neighbor of a and b[7]. Since $\triangle abc$ satisfies that $\odot abc$ does not contain any 2-hop neighbor of a, b, or c, both e and d are outside $\odot abc$. Let ed intersect with $\odot abc$ at e' and d'. Let o be the center of $\odot abc$. There are two cases based on the location of ed.

Case 1: ed and b are on the same side of o, as shown in Figure 2.7. It is easy to see that $\angle ebd > \angle e'bd' > \pi/2$. Therefore, $|bd| < |ed| \leq R$, which contradicts the assumption that $|bd| > R$.

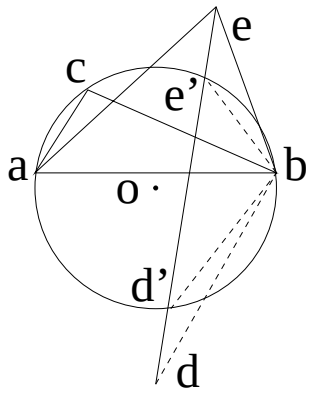


Figure 2.7: ed and b are on the same side of o

Case 2: ed and a are on the same side of o, or o sits exactly on ed. Based on the same rationale as in Case 1, it is easy to see that $|ad| < |ed| \leq R$, which contradicts the assumption $|ad| > R$. Thus, the lemma follows. □

Lemma 7 *Each edge of a 2-localized Delaunay triangle belongs to* $E - CE$.

Proof: Based on Lemma 5 and Lemma 6, each edge of a *2-localized Delaunay triangle* is not a *Constrained edge* (does not belong to *CE*). Since each edge of a *2-localized Delaunay triangle* belongs to E, the lemma follows. □

[7] Nodes that can be reached by node a *within* (less than or equal to) k hops are called k-hop neighbors of a. This implies that all k-hop neighbors of a node are also $(k+1)$-hop neighbors of the node.

Lemma 8 *Each 2-localized Delaunay triangle is an ECLDel$\triangle$.*

Proof: A *2-localized Delaunay triangle*, say $\triangle abc$, satisfies that the interior of its circumcircle does not contain any 2-hop neighbor of a, b, or c. Based on the definition of k-hop neighbors of a node described in Section 2.3.4, it is easy to see that the set of $(k+1)$-hop neighbors of a node contains all the k-hop neighbors of the node. This implies that the set of 2-hop neighbors of a node contains all 1-hop neighbors of the node. Therefore, the interior of the circumcircle of $\triangle abc$ does not contain any 1-hop neighbors of a, b, or c.

Since each edge of $\triangle abc$ belongs to $E-CE$, $\triangle abc$ satisfies the definition of a *ECLDel$\triangle$*. Thus, the lemma. □

Theorem 1 *LDel$^{(2)}$(N) is a subgraph of ECLDel(N).*

Proof: We know that *ECLDel(N)* contains all *Gabriel edges* and all *ECLDel$\triangle$*s. *LDel$^{(2)}$(N)* contains all *Gabriel edges* and all *2-localized Delaunay triangles*. The theorem follows from Lemma 8. □

Since *LDel$^{(2)}$(N)* is a t-spanner of *UDG(N, E)* and a subgraph of *ECLDel(N)*, *ECLDel(N)* is a t-spanner of *UDG(N, E)*.

2.4.4 ECLDel is Planar

In this section, we prove that the *ECLDel* is a planar graph. Recall that the *1-localized Delaunay graph*, *LDel$^{(1)}$(N)*, contains all *Gabriel edges* and all *1-localized Delaunay triangles* [42]. A *1-localized Delaunay triangle*, say $\triangle abc$, satisfies that the interior of $\odot abc$ does not contain any 1-hop neighbor of a, b, or c, and each edge of $\triangle abc$ has a length no more than R (belongs to E).

By showing that each *ECLDel$\triangle$* belongs to the set of all *1-localized Delaunay triangles*, we know that each *ECLDel$\triangle$* satisfies the properties of the *1-localized Delaunay triangles*, and thus, we prove that *ECLDel* is planar.

Lemma 9 *Each ECLDel$\triangle$ belongs to the set of all 1-localized Delaunay triangles.*

Proof: Definition 4 describes an *ECLDel*$\triangle$, say $\triangle abc$, which satisfies that the interior of $\odot abc$ does not contain any 1-hop neighbors of a, b, or c, and each edge of $\triangle abc$ belongs to $E - CE \subseteq E$. Therefore, an *ECLDel*$\triangle$ must be a *1-localized Delaunay triangle*. The lemma follows. □

Theorem 2 *No two ECLDel*$\triangle$*s intersect.*

Proof: Suppose two *ECLDel*$\triangle$s $\triangle abc$ and $\triangle def$ intersect. Lemma 9 implies that the *ECLDel*$\triangle$s satisfy the properties of the *1-localized Delaunay triangles*. The only way this can happen is if exactly one edge of each triangle is not intersected by the edges of the other triangle [42].

Suppose ac and ef are not intersected by $\triangle def$ and $\triangle abc$, respectively, then ab and bc intersect with both de and df, as shown in Figure 2.8. Either $\odot abc$ contains at least one of the nodes d, e, and f, or $\odot def$ contains at least one of the nodes a, b, and c [42]. Let us suppose that $\odot abc$ contains d as shown in Figure 2.8.

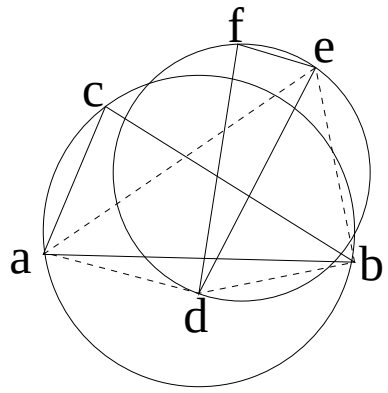


Figure 2.8: A case in which two *Edge Constrained Localized Delaunay triangles* intersect

Since $\odot abc$ does not contain any 1-hop neighbors of a, b, or c, $ad > R$, $db > R$ and $cd > R$. For the intersecting edges ab and de, at least one of the nodes a, b, d and e is the 1-hop neighbor of the other three [19]. Therefore, only e is the 1-hop neighbor of the other three nodes, which means that $ae \leq R$ and $eb \leq R$.

In cases where de is not a *Gabriel edge*, then ab is an *Unaware Intersection (UI) edge* from Definition 2. In cases where de is a *Gabriel edge*, ab is an *Intersecting Gabriel (IG) edge* from Definition 1. Either case implies that ab is a *Constrained edge*, which contradicts that each edge of a *ECLDel*$\triangle$ is not a *Constrained edge*, (belongs to $CE = E - CE$). Hence, the theorem. □

Theorem 3 *ECLDel(N) is planar.*

Proof: ECLDel(N) contains all *Gabriel edges* and all *ECLDel*$\triangle$s. *Gabriel edges* do not intersect with each other. *ECLDel*$\triangle$s do not intersect with each other based on Theorem 2. Each edge of an *ECLDel*$\triangle$ is not an *Intersecting Gabriel (IG) edge*, so a *Gabriel edge* does not intersect with any *ECLDel*$\triangle$ edges. Hence, the theorem. □

2.4.5 A Comparison of Graphs ECLDel and PLDel

A common feature of *ECLDel* and *PLDel* is that both of them are planar t-spanners of *UDG*. However, they are not the same, as *ECLDel* is a subgraph of *PLDel*. We present a proof of this below.

Lemma 10 *ECLDel is a subgraph of LDel*$^{(1)}$*(N).*

Proof: Based on Definition 4, an *ECLDel* $\triangle$ satisfies that its interior does not contain any 1-hop neighbors of any of its three vertices, and each of its edges belongs to the edge set $E - CE \subseteq E$. This means that each *ECLDel* $\triangle$ is a 1-localized *Delaunay* triangle, *LDel*$^{(1)}\triangle$. Based on the definition of *LDel*$^{(1)}$*(N)* [42] and Definition 5, the lemma follows. □

Theorem 4 *ECLDel is a subgraph of PLDel.*

Proof: To construct a *PLDel*, the intersections in *LDel*$^{(1)}$*(N)* should be removed. Because the *Gabriel* edges do not intersect with each other, the intersections in *LDel*$^{(1)}$*(N)* have two cases. First, a *LDel*$^{(1)}\triangle$ intersects with a *Gabriel* edge, which must contain an

IG edge. In this case, the *LDel*$^{(1)}\triangle$ will be removed. Secondly, two *LDel*$^{(1)}\triangle$*s*, say $\triangle abc$ and $\triangle xyz$, intersect. We have that exact two edges of each of the triangles intersect with each other, and either $\odot abc$ contains at least one of the nodes x, y and z or $\odot xyz$ contains at least one of the nodes a, b and c [42]. Suppose the intersection is as in Fig 2.9 below, where ab and bc intersect with xy and xz, and x is inside $\odot abc$. In this case, $\triangle abc$ will be removed. Because $|ax| > R, |bx| > R, |az| \leq R$ and $|bz| \leq R$ [42], ab is an UI edge. Therefore, to construct a *PLDel*, each *LDel*$^{(1)}\triangle$ that is removed from the *LDel*$^{(1)}$*(N)* contains either an IG or an UI edge, which is not contained in the *ECLDel*. Based on the definition of *LDel*$^{(1)}$*(N)* [42], Definition 5, and Lemma 10, the theorem follows. □

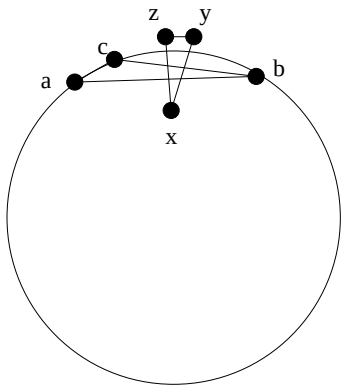


Figure 2.9: An example of the intersection of two *1-localized Delaunay triangles*.

On the contrary, *PLDel* is not a subgraph of *ECLDel*, because some triangles within may contain UI edges that are not contained in *ECLDel*.

2.5 An Algorithm to Construct an ECLDel

In this section, we present algorithm *AlgEcldel*, with which each node constructs the *ECLDel* distributively with 1-hop neighborhood information. $N(a)$ denotes the set of node a's 1-hop neighbors, including a.

2.5.1 Algorithm AlgEcldel

The details of algorithm *AlgEcldel* are as follows:

1. Each node gets the location information of its 1-hop neighbors from a node's periodic broadcasting of Hello messages.

2. A node, say a, computes the *Delaunay triangulation, Del(N(a)),* which can be computed via a variety of methods [15, 17, 22].

3. Node a finds all *Unaware Intersection (UI) edges* it can discover and inserts them into set *UI* of *Unaware Intersection edges*. Node a finds all *Gabriel edges* incident on it, marks them as edges in the *ECLDel*, and inserts them to set *GE* of *Gabriel edges*. Node a **broadcasts** a message, *edges(UI, GE)*.

4. On receiving the message *edges(UI, GE)* from the nodes in $N(a)$, a combines the edges in sets *UI* it receives with its own set *UI*, and combines the edges in sets *GE* it receives with its own set *GE*.

5. Node a selects all triangles incident on it from *Del(N(a)),* each edge of which satisfies that it has a length no more than R, does not belong to set *UI*, and does not intersect with any edge in set *GE* (not an *IG* edge). The triangles are candidates for the *ECLDel*$\triangle$s. Node a puts all the candidate triangles in the message *candidates*, e.g., *candidates((*a*,* b*,* c*), (*a*,* e*,* f*))*, and **broadcasts** the message.

6. Node a will keep a candidate triangle $\triangle abc$ as an *ECLDel*$\triangle$ if it receives *candidates((*a*,* b*,* c*))* from both b and c. Node a marks ab and ac as edges in the *ECLDel*.

By locally applying the algorithm *AlgEcldel*, each node computes all *Gabriel edges* and all *ECLDel*$\triangle$s incident on it distributively, to construct the *ECLDel* graph.

2.5.2 Correctness Proof of Algorithm AlgEcldel

In this section, we prove the correctness of the algorithm *AlgEcldel*.

Theorem 5 *Algorithm* AlgEcldel *constructs a graph of* ECLDel.

Proof: Suppose the algorithm *AlgEcldel* constructs a graph G. From step 2 of *AlgEcldel*, a node, say a, computes all the *Delaunay triangles* incident on it, whose circumcircles do not contain any 1-hop neighbors of node a. From step 3 and step 4, node a knows all the UI edges incident on it and all the *Gabriel* edges incident on its 1-hop neighbors based on Corollary 2. From step 5, node a knows all the IG edges incident on it in the *Delaunay triangles* it computes based on Corollary 1. With the knowledge of UI and IG edges, node a selects the candidate *ECLDel*$\triangle$*s*. For example, if $\triangle abc$ is a candidate *ECLDel*$\triangle$ selected by node a, then it satisfies that $ab \leq R, bc \leq R$, and each of the edges ab and ac is neither an UI nor an IG edge. Therefore, if $\triangle abc$ is selected as a candidate *ECLDel*$\triangle$ by all nodes a, b and c, then it satisfies the following:

1. n, p and q are not inside $\odot abc$, $\forall n \in N(a), \forall p \in N(b), \forall q \in N(c)$;
2. $ab \leq R, ac \leq R$ and $bc \leq R$;
3. Each of the edges ab, bc and ac is neither an UI nor an IG edge.

Based on Definition 4, $\triangle abc$ is an *ECLDel*$\triangle$. Node a keeps each edge of all such *ECLDel*$\triangle$*s* incident on it in graph G from step 6. Node a also keeps all the *Gabriel* edges incident on it in graph G as in step 3. Based on Definition 5, graph G is an *ECLDel* graph. Hence, the theorem. $\square$.

2.5.3 Communication Complexity of Algorithm AlgEcldel

Let n denote the number of nodes in the network. Then each node's identity needs $\log n$ bits to be expressed. Therefore, for each node, the communication cost for broadcasting messages is $O(\log n)$ bits. As a result, the total communication cost in the network is $O(n \log n)$ bits.

2.5.4 A Comparison of the Algorithms to Construct an ECLDel and a PLDel

For simplicity, the algorithm to construct a *PLDel* is denoted by *AlgPldel*. A common feature of *AlgEcldel* and *AlgPldel* is that both can be run by each node independently with 1-hop neighborhood information. In addition, they have the same asymptotic communication complexities, $O(n \log n)$.

However, the upper bounds of the communication complexities of both algorithms are within different constant factors. Let $f_{ECLDel}(n)$ and $f_{PLDel}(n)$ denote the communication complexities of *AlgEcldel* and *AlgPldel*, respectively. Then $f_{ECLDel}(n) \leq e * n \log n$ and $f_{PLDel}(n) \leq p * n \log n$, where e and p are two positive constant factors. The difference is that e is much smaller than p. This is for two reasons. First, the *Intersecting Gabriel (IG) edges* and the *Unaware Intersection (UI) edges*, which are constrained in the *ECLDel* graph, help significantly reduce the number of candidate triangles, and as a result, reduce the size of messages broadcast by each node. Second, the number of messages broadcast by each node is significantly reduced from five rounds (each round may contain several messages) to two messages.

A decrease in both the number and the size of messages broadcast by each node reduces the communication cost, and saves the network bandwidth and node power, which is desirable for mobile ad hoc and sensor networks. Our simulation results show that the average number of messages and the average size of messages broadcast by each node is, respectively, 65% and 42% less in the construction of *ECLDel* than that in *PLDel*, which supports the above analysis.

2.6 Simulation Study and Analysis

2.6.1 Simulation Settings

In our simulations, the network is a square area of $1000 \times 1000m^2$. Nodes are distributed randomly in the area and have a transmission range 200 meters. Nodes construct a *unit-disk*

graph, where there is an edge between two nodes if they are within each other's transmission range. The number of nodes in the network is varied from 20 to 240. By changing the number of nodes in the network, we change the network density or the average node degree (the number of neighbors of a node). We do not simulate the network with less than 20 nodes, in which case the node degree is less than 2.5, because the network would be highly likely to be partitioned.

In the simulation study, we generate seventeen network topologies randomly, which contain $20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 140, 160, 180, 200, 220$ and 240 nodes. For each topology, we ran simulations 20 times with different seeds. Each value in the following graphs is the average of the 20 runs.

2.6.2 Performance of GFG Geographic Routing on Different Underlying Graphs

In this section, we evaluate the performance of Greedy-Face-Greedy (GFG) geographic routing on four underlying graphs, which are *RNG*, *GG*, *PLDel*, and *ECLDel*. *ECLDel* and *PLDel* are planar t-spanners of *UDG*, which are denser than *RNG* and *GG*. Therefore, the geographic routing on *ECLDel* and *PLDel* should perform better with shorter routes than that on *RNG* and *GG*. Our simulation results, discussed below, confirm this.

We use GPSR [30] as the GFG geographic routing protocol, in which each source node takes greedy routing first, face routing when the greedy routing fails, and then greedy routing again (if possible) to its destination. Greedy routing is applied on the original *UDG*. We randomly select 10% nodes as source nodes and for each we randomly select 10% nodes as destination nodes. We depict the *success rate* of greedy routing on *UDG* in Figure 2.10.

Figure 2.10 illustrates that when the number of nodes in the network is 140, the *success rate* of greedy routing on *UDG* is 99.64%, which is close to 100%. However, when the number of nodes is less than 140, the *success rate* of greedy routing on *UDG* is less than 100%. This implies that greedy routing may fail and face routing is required.

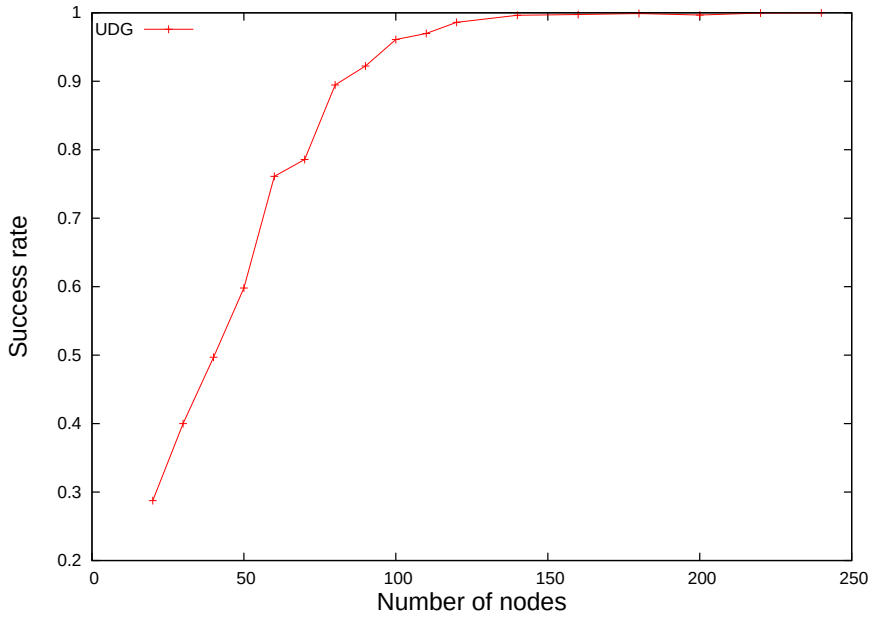


Figure 2.10: *Success rate* of greedy routing on *UDG*

Figure 2.11 shows an example of network graphs of *UDG*, *Del*, *RNG*, *GG*, *PLDel*, and *ECLDel*, where 80 nodes are distributed randomly in the network. *Del* graph is not suitable for mobile ad hoc networks, because it needs global information and may have edges with length more than the transmission range of mobile nodes. *RNG*, *GG*, *PLDel*, and *ECLDel* are planar graphs, among which *PLDel* and *ECLDel* are t-spanners of *UDG*.

We evaluate the performance of GPSR, the greedy routing of which is applied on the original *UDG* and the face routing of which is applied on four planar underlying graphs, *RNG*, *GG*, *PLDel*, and *ECLDel*, respectively. We randomly select 10% nodes as source nodes and for each of them we randomly select 10% nodes as destination nodes. We show the average path length (hop count) for GPSR on the four graphs in Figure 2.12.

Figure 2.12 implies that GPSR with face routing applied on *ECLDel* and *PLDel* outperforms that applied on *GG* and *RNG* in path length (hop count). This is because *ECLDel* and *PLDel* are planar t-spanners of *UDG*, which are denser than *RNG* and *GG*. Note that neither *GG* nor *RNG* is a t-spanner of *UDG*.

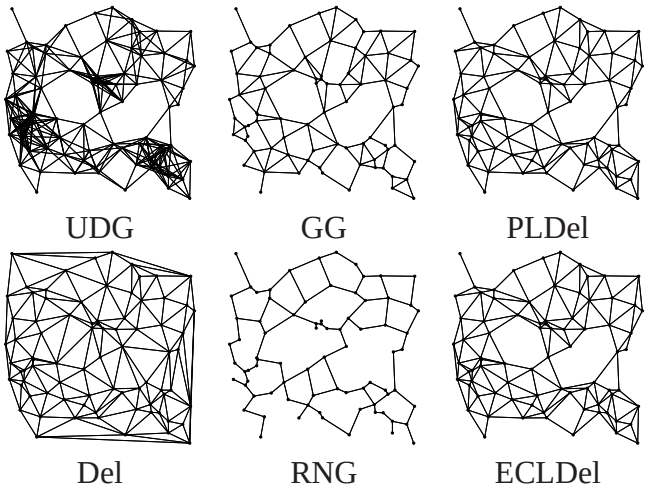


Figure 2.11: Examples of network topologies

2.6.3 Evaluation of the Cost of Construction of an ECLDel and a PLDel

In this section, we evaluate the communication cost for the construction of *PLDel* and *ECLDel* using the following three metrics: the average number of messages broadcast by each node, the total number of messages broadcast in the network, and the average size of messages broadcast by each node.

The average number of messages broadcast by each node

Figure 2.13 shows the average number of messages broadcast by each node, which is 65% less in construction of *ECLDel* than that of *PLDel*. This is due to the fact that each node puts all the candidate triangles in one broadcast message in our algorithm, instead of in different broadcast messages as in the algorithm to construct *PLDel*.

Total number of messages broadcast in the network

Figure 2.14 shows the total number of messages broadcast in the network, which is 69% less in the construction of *ECLDel* than that of *PLDel*.

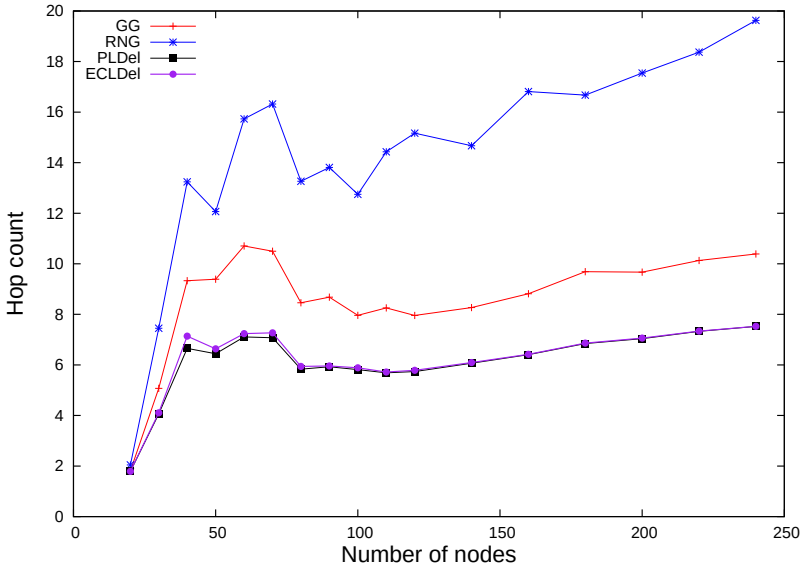


Figure 2.12: Hop counts of GPSR on different underlying graphs

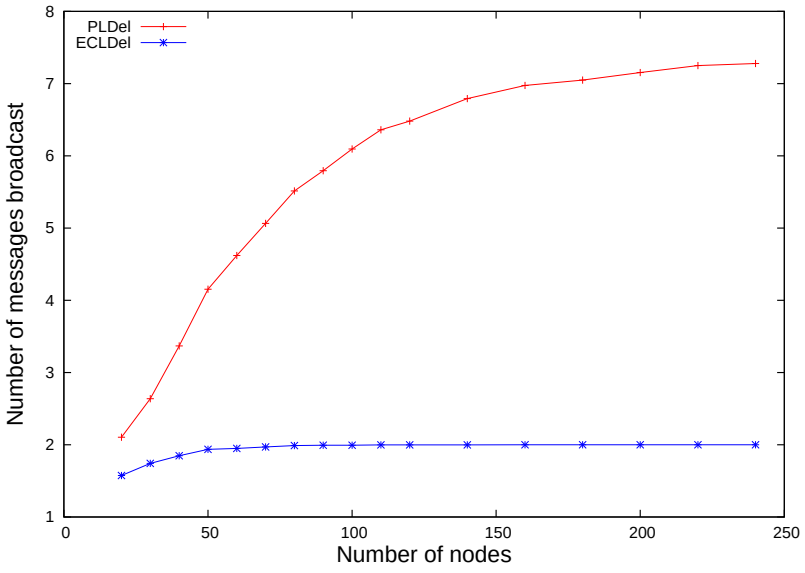


Figure 2.13: Average number of messages broadcast by each node

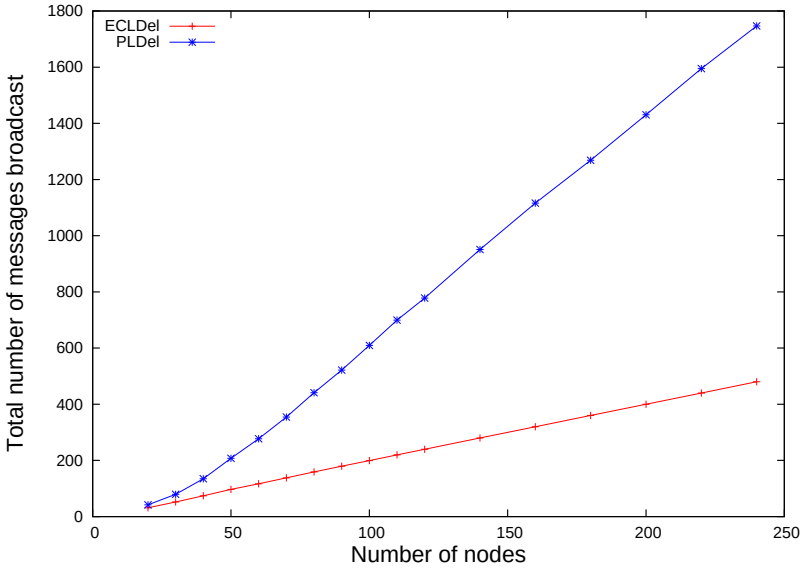


Figure 2.14: Total number of messages broadcast in the network

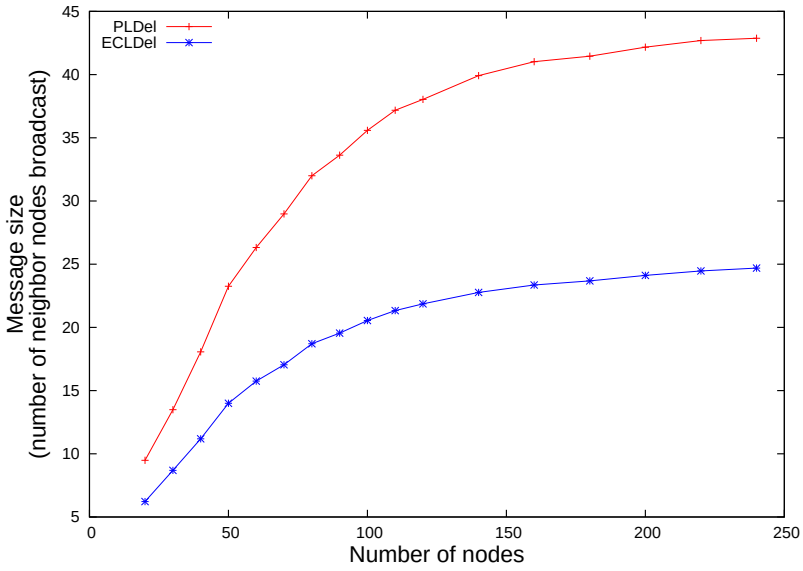


Figure 2.15: Average number of neighbor nodes in messages broadcast by each node

The average size of messages broadcast by each node

We depict the average size of messages in terms of the number of nodes in the messages broadcast by each node in Figure 2.15. The average number of neighbor nodes for each node to broadcast in a message is 42% less in the construction of *ECLDel* than that in that of *PLDel*. This is due to the fact that by constraining the *Intersecting Gabriel (IG) edges* and the *Unaware Intersection (UI) edges* in the construction of *ECLDel*, each node gets far fewer candidate triangles, which results in a much smaller size of messages, to broadcast.

In the construction of *ECLDel*, besides the candidate triangles, each node also needs to broadcast the *UI* edges discovered by it. However, Figure 2.16 shows that the average number of *UI* edges broadcast by each node is very low, 0.12 on average.

Communication cost of construction of *PLDel* and *ECLDel*

We depict the communication cost in terms of the number of neighbor nodes in the messages broadcast in the network, with respect to the number of nodes in Figure 2.17. It shows that the communication cost in construction of *ECLDel* is 83% less than that of *PLDel*. This confirms that the algorithm to construct *ECLDel* is more desirable for mobile ad hoc and sensor networks.

2.7 Related Work

Gabriel graph (GG) [18] and *relative neighborhood graph (RNG)* [62] are commonly used as the underlying graphs for face routing. For instance, *GG* is used by Bose et al. [10] and *RNG* is used by Karp et al. [30]. However, both *GG* and *RNG* are relatively sparse, and neither is a t-spanner of *UDG*, which results in long routes in face routing on them.

Boone et al. [9] try to construct a planar spanning subgraph of *UDG* denser than *GG* by doing some extra tests, which may decrease the path length in face routing. However, their improvements are minor in dense networks, and the graph they constructed is not a t-spanner.

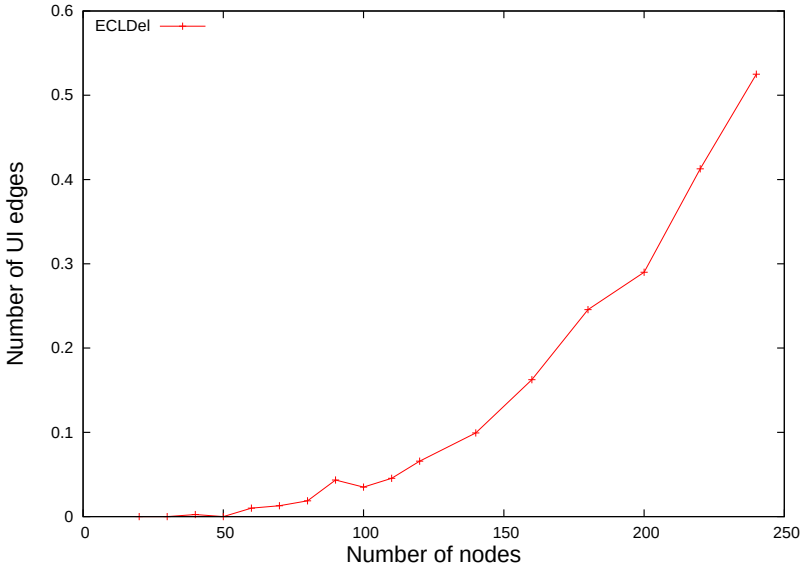


Figure 2.16: Average number of *UI* edges broadcast by each node

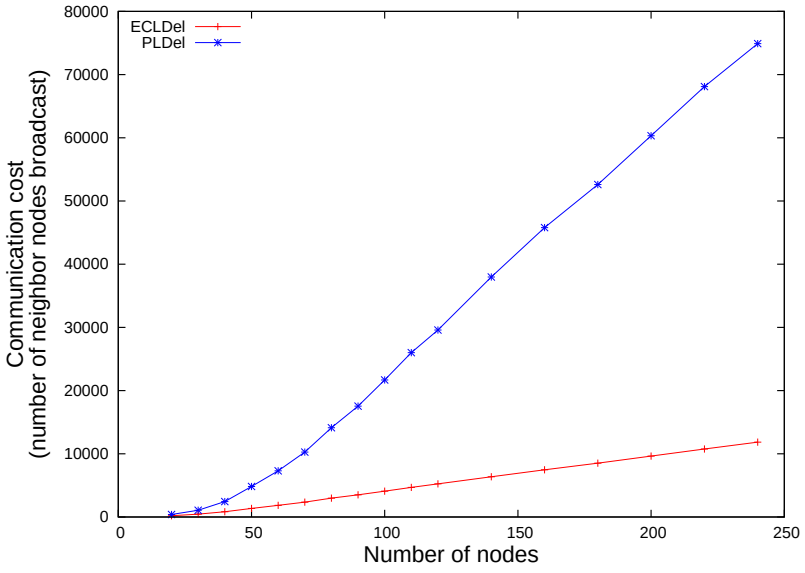


Figure 2.17: Communication cost in construction of *ECLDel* and *PLDel*

Hu [24] constructs a planar graph as the network topology that has a bounded node degree. To decide whether an edge ab belongs to the graph, a check is required to determine whether a circle passes through a and b without any other node in its interior. This means Hu's method does not converge in the worst case. Furthermore, the network topology graph may not be a t-spanner.

Both the Yao graph [68] and θ-graph [25] have been proved to be t-spanners, but they may not be planar graphs.

Several methods [40,41,67] to construct wireless network topologies that are t-spanners have been proposed. However, none of them is guaranteed to be a planar graph.

Delaunay triangulation [50] is a planar t-spanner of the completed Euclidean graph, which is almost as good as the complete graph [14]. However, it is hard to construct because it needs global information and is not suitable for mobile ad hoc or sensor networks as some edges in it may have a greater length than the transmission range of mobile nodes.

Gao et al. [19] propose a *restricted Delaunay graph*, *RDG*, as the underlying graph for geographic routing protocols, which is a planar t-spanner of the original *UDG*. In their method, the network is divided into clusters, each of which has a clusterhead. Clusters are connected by gateway nodes (gateways). Their graph contains all edges between each node and its clusterhead, and a planar graph among clusterheads and gateways. The communication cost of this method may be $\Theta(n^2)$, and the computation cost may be $\Theta(n^3)$ in the worst case, where n is the number of nodes in the network.

Li et al. [42] propose a *planarized localized Delaunay graph*, *PLDel*, as the underlying graph for geographic routing, which is a planar t-spanner of *UDG* and can be constructed by each node distributively. The communication cost in constructing *PLDel* is $O(n \lg n)$ and the computation cost in constructing *PLDel* is $O(d \lg d)$, where d is the average node degree in the network, which is near optimal. However, the construction of *PLDel* is not efficient enough because each node needs to broadcast several messages, which makes their algorithm converge slowly.

Another algorithm [5] tries to decrease the number of broadcast messages needed in the construction of *PLDel*. However, inconsistent *Delaunay triangles* may exist among nodes.

Wang et al. [65] propose a bounded degree planar t-spanner of *UDG*, which plugs in the work of Li et al. [42], and whose construction needs 2-hop neighborhood information of nodes.

2.8 Chapter Summary

In mobile ad hoc and sensor networks, most geographic routing protocols, e.g., Greedy-Face-Greedy routing protocols, need nodes to construct planar graphs as the underlying graph for face routing. Li et al. [42] proposed a planar t-spanner of *UDG*, called *planarized localized Delaunay graph, PLDel*, for geographic routing. However, their algorithm to construct the *PLDel* is highly complex and converges slowly, as each node needs to broadcast too many messages, which results in high communication cost.

In this chapter, we proposed an *Edge Constrained Localized Delaunay graph, ECLDel*, as the underlying graph for geographic routing in mobile ad hoc and sensor networks. We proved that the *ECLDel* is a planar t-spanner of the original *unit-disk* graph. We developed an algorithm $AlgEcldel$ to construct the *ECLDel*, which can be run by each node distributively with 1-hop neighborhood information. Compared to the algorithm to construct the *PLDel*, our algorithm to construct the *ECLDel* is much simpler and converges faster. This is due to the fact that we significantly decrease the number of messages and the size of messages broadcast by each node in the construction, which results in a much lower communication cost and is more desirable for mobile ad hoc and sensor networks.

Our simulation results confirm this, and the average number of messages and the average size of messages broadcast by each node are, respectively, 65% and 42% less under our algorithm than in the algorithm to construct the *PLDel*.

Chapter 3

A Pre-Processed Cross Link Detection Protocol for Geographic Routing in MANET under Realistic Environments with Obstacles

3.1 Introduction

In mobile ad hoc and sensor networks, a variety of geographic routing protocols have been developed [10, 21, 26, 30, 34–36, 43, 59, 64]. Among them, *GFG* geographic routing protocols [30, 35, 36] have been actively researched in recent years. Most of them make the idealized assumption that all mobile nodes in the network have the same transmission range and construct a *unit-disk graph*, *UDG*, in which there is an edge incident on two nodes, if and only if the *Euclidean* distance between them is no more than the transmission range. Based on this assumption, many planarization algorithms are used to construct planar graphs for face routing [18, 19, 42, 50, 60, 62, 65].

However, in realistic environments, the assumption of *UDG* may be violated in the following three situations:

1. Obstacles may exist between two neighboring nodes, and hence an edge (communication link) may not exist between these even though the *Euclidean* distance between them is less than the transmission range. We address this problem in this chapter.

2. Nodes with different power may have different transmission ranges, making links between such nodes unidirectional.

3. The location information obtained by *GPS* or other systems may be inaccurate and are error prone. Seada et al. [56] observe that many state-of-the-art techniques usually cause 10% (of the transmission range) or more in location error, which leads to incorrect calculation of the *Euclidean* distance between two nodes.

Violating the *UDG* assumption causes the idealized planarization algorithms for *UDG* to not work correctly, causing breakdown of face routing. Usually, there are three situations caused by the idealized planarization algorithms:

1. Crossing edges may exist in the subgraph after planarization.

2. The subgraph may not be connected due to removal of edges that should not be removed.

3. The two nodes that an edge is incident on do not agree on the existence of the edge.

Kim et al. [32] implement the geographic routing protocol, *GPSR*, in realistic environments with obstacles. In one of their testbeds, they show that over 30% of node pairs can not find a path to each other by *GPSR*, as the face routing of *GPSR* is applied on the *relative neighborhood graph* [62], *RNG*, which is extracted from the idealized network graph *UDG*.

Authors of other research studies [28,29,56] propose a mutual witness technique, where two nodes at the two ends of a link discuss and make an agreement with each other on whether to keep the link or not. However, this still does not preclude the three situations described above.

3.1.1 Research Motivations

Kim et al. [32] propose a *Cross Link Detection Protocol (CLDP)*, which to our knowledge is the only one that can make *GFG* geographic routing protocols, like *GPSR*, work correctly

under realistic conditions with obstacles. They produce a spanning subgraph of the original realistic network graph, on which face routing can work correctly.

In *CLDP*, a node needs to probe each link attached to it in the original realistic network graph to detect cross-links [1]. If a node detects cross-links, it may need further probing of a link to decide if the link should be kept or removed.

The probing of a link requires traversal of the network graph using the *Right-Hand Rule (RHR)*, which in some cases may lead to a long face with a large number of edges to traverse. This results in a high communication cost because each traversal of an edge needs a message broadcast. In addition, for some of its links, a node may need several rounds of probing (several rounds of traversal of faces) to make a decision, which makes their method converge slowly.

This challenge drove our development of a cross link detection protocol, which makes the *GFG* geographic routing work efficiently and correctly under realistic environments with obstacles.

3.1.2 Main Contributions

In this chapter, we propose a *Pre-Processed Cross Link Detection Protocol (PPCLDP)*. It generates an almost planar spanning subgraph[2] of the original network graph, on which the *GFG* geographic routing can work correctly in realistic environments with obstacles.

The proposed *PPCLDP* improves *CLDP* by adding a *2-hop Cross Link Pre-Processing (CLPP)* algorithm. In the *CLPP* algorithm, a node can detect *2-hop cross links* of any links attached to it and can decide whether to keep or remove the links by exchanging a few messages with its neighbors. In this way, a node does not need to probe these links using the *Right-Hand Rule (RHR)*, which may cause long faces traversed, and does not need several rounds of probing for them. This, decreases the total number of messages

[1] Two links intersecting are called cross-links.

[2] Some cross-links may exist in the subgraph, because their removal may cause partition of the network under realistic environments.

broadcast by nodes significantly, which makes *PPCLDP* converge faster than *CLDP* with much less communication cost.

Section 3.2 below presents preliminaries, while Section 3.3 describes the proposed *PP-CLDP*. Section 3.4 presents the results of a simulation study and analysis. Section 3.5 describes the related work. Section 3.6 concludes this chapter.

3.2 Preliminaries

Before presenting our *PPCLDP* work, we introduce the previous *CLDP* work as a preliminary.

Cross Link Detection Protocol (CLDP)

Kim et al. proposed a *Cross Link Detection Protocol (CLDP)* [32], which generates an almost planar subgraph of the original network graph under realistic environments with obstacles, on which *GFG* routing works correctly.

In *CLDP*, each link in the original graph is initially marked as *routable* [3] and is probed to see whether to keep or remove it. When *CLDP* decides to remove a link, it will be marked as *non-routable*. The set of *routable* links form a *routable* subgraph. All probe messages in *CLDP* traverse the current snapshot of the *routable* subgraph. The probing stops when further probing of links would not cause any link to be removed. When the probing stops, all the *routable* links form the final subgraph for face routing. *CLDP* involves the following steps:

1. Step 1. Detect cross-links:

   A node probes each link attached to it in the original network graph to see if it intersects with other links. This is achieved in the following way:

   A node, say a, puts the location information of both ends of a probed link in a probe message, which will traverse the current *routable* network graph by the *Right-Hand*

[3] For a link, say ab, if it can be used as part of a path in the routing, then it is called *routable*.

Rule (RHR). If there is a link intersecting with the probed link, one end node of the cross link may know the intersection and add the location of the end nodes of the cross link in the probe message. When the probe message returns to node a, a will know if a cross link of the probed link exists.

2. Step 2. Decide whether to keep or remove the cross-links:

   If node a finds that there is a cross link, say cd, of the probed link, say ab, node a will check whether ab or cd can be removed.

   If a link is traversed by a probe message twice (once in each direction), then *CLDP* treats this link as non-removable. Otherwise, the link is considered to be removable. Based on this rule, node a knows whether link ab or cd is removable or not by checking whether they are traversed by the probe message twice or not. The following four cases arise:

   Case 1 : Both ab and cd can be removed (i.e., the probe message traverses neither link twice).

   Case 2 : ab can be removed, but cd can not be removed.

   Case 3 : ab can not be removed, but cd can be removed.

   Case 4 : Neither ab nor cd can be removed.

   For Case 1 and Case 2, node a will begin a second-phase probing and send a commit message to decide whether to remove the probed link ab or not. Note that a can not remove link ab directly, and the second phase probing is to avoid network partition. The commit message will traverse the current *routable* graph by *RHR*. If the commit message succeeds (i.e., the commit message returns to a), a will remove link ab by marking it as *non-routable* and notify node b about the removal of ab. In addition, the two adjacent links, which are obtained by applying both the *Right-Hand Rule (RHR)* and *Left-Hand Rule (LHR)* to link ab, will be re-probed.

For Case 3, node a will begin a second-phase probing and send a commit message to confirm whether link cd can be removed. If the commit message succeeds (i.e., the commit message returns to a), a will notify both nodes c and d to remove link cd (to mark cd as *non-routable*). In addition, a will re-probe link ab, because the removal of cd may cause a to find new cross links of ab.

For Case 4, a will do nothing.

It is easy to see that in *CLDP*, each node needs to probe each of its links for one or more rounds to detect whether it intersects with other links and to decide whether it needs to be removed. The probing of a link traverses the network graph using the *Right-Hand Rule (RHR)*, which in some cases may lead to a long face with a large number of edges to traverse. In addition, for some of its links, a node may need several rounds of probing (several rounds of traversal of faces) to make a decision. This results in a high communication cost and is time consuming. Next, we present a far more efficient cross link detection protocol.

3.3 *Pre-Processed Cross Link Detection Protocol (PPCLDP)*

The proposed *Pre-Processed Cross Link Detection Protocol, PPCLDP*, generates an almost planar subgraph of the original network graph, which makes face routing work correctly and as a result, means that *GFG* geographic routing never fails under realistic environments with obstacles.

The proposed *PPCLDP* contains a *2-hop Cross Link Pre-Processing (CLPP)* algorithm and a *Restricted Cross Link Detection Protocol (RCLDP)*. Before presenting the details of these two, we introduce assumptions.

3.3.1 Assumptions

In this chapter, we are concerned with the mobile ad hoc and sensor networks under realistic environments with obstacles. Therefore, we make the following assumptions:

1. The original network graph is connected.
2. Each node knows its own position by *GPS* or other positioning services [12, 23]. We do not consider the position error as in *CLDP* [31, 32].
3. By broadcasting its own location, each node knows all its neighbors' locations, with which it constructs a *link set* that contains all links between itself and its neighbors.
4. By broadcasting the *link set* of itself, each node knows the *link sets* of all its neighbors.
5. There is no unidirectional link in the network.

3.3.2 2-hop Cross Link Pre-Processing (CLPP) Algorithm

In this section, we present the *2-hop Cross Link Pre-Processing (CLPP)* algorithm, in which a node can detect any *2-hop cross links* of a link attached to it and can decide whether to keep or remove the link by exchanging a few messages with its neighbors. The graph generated by the *CLPP* algorithm is called the *CLPP* graph. We prove that it is a connected graph. Next, we introduce a few definitions.

Definition 6 *For node a, the set that contains links in the link sets of all its neighbors is called the 2-hop link set of a.*

Definition 7 *For link ab that is attached to a, the link that is crossing ab and is in the 2-hop link set of a, is called a 2-hop cross link of ab detected by a.*

Corollary 4 *If node a detects a 2-hop cross link of ab, say link cd, then either c or d or both c and d are neighbors of a.*

Definition 8 *If link ab is crossing with cd, then the link pair of ab and cd is called a cross link pair.*

Definition 9 *If link cd is a 2-hop cross link of ab detected by either a or b, then the link pair of ab and cd is called a 2-hop cross link pair.*

Definition 10 *The status of a link, which includes existing/non-existing, and routable/non-routable, is called the link status.*

Our simulation results show that the ratio of the average number of *2-hop cross link pairs* to that of *cross link pairs* is 82% in a *rough* environment, where the number of obstacles is the same as the number of nodes. This motivated us to develop the *CLPP* algorithm that pre-processes the *2-hop cross link pairs*.

The CLPP Algorithm

Each link in the original network graph is marked as *routable* initially. At the network start-up, each node implements the *CLPP* algorithm distributively, which makes a node decide whether to keep or remove a link attached to it that has *2-hop cross link(s)* detected by it. For node a, the details of the algorithm *CLPP* are as follows:

1. For link ab that is attached to it, node a checks if there is one or more *2-hop cross links* of ab detected by it based on the *2-hop link set* of itself.

2. If a does not detect any *2-hop cross links* of ab, node a will keep ab as *routable*.

3. Otherwise, let cd denote a *2-hop cross link* of ab detected by a. Node a checks whether ab can be removed because of cd as follows:

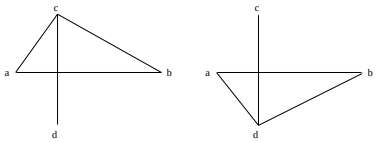
    i. Based on Corollary 4 and the assumption that each node knows the *link sets* of all its neighbors, node a knows the connectivity of nodes a, b, c and d. With this, node a checks whether there are one or more loops, which are formed by

nodes among a, b, c and d, and contain link ab. Such a loop(s) is denoted by $L(ab, cd)$.

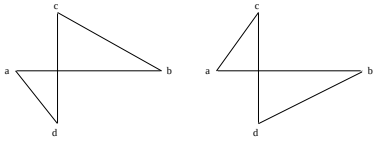
ii. If $L(ab, cd)$ does not exist, then ab can not be removed because of cd.

iii. If only one loop $L(ab, cd)$ exists, node a will check if ab can be removed due to the loop $L(ab, cd)$.

Figure 3.1 (a), (b), (c) and (d) shows all the four cases of the loop $L(ab, cd)$, which equal to the loop abc, abd, $abcd$ and $abdc$, respectively.



(a) Loop abc (b) Loop abd



(c) Loop $abcd$ (d) Loop $abdc$

Figure 3.1: Four cases of the loop $L(ab, cd)$

For each of the four cases shown in Figure 3.1, node a runs an algorithm *IfRemove(ab, cd, L(ab, cd))*, the details of which will be discussed later.

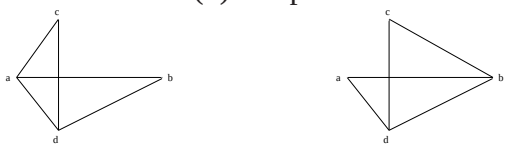
If the return value of the *IfRemove* algorithm is *True,* ab can be removed due to the loop $L(ab, cd)$. Otherwise, ab can not be removed because of cd.

iv. If $L(ab, cd)$ contains more than one loop, node a will further check if ab can be removed due to any of these loops.

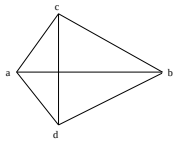
Figure 3.2 shows all five cases of the loops $L(ab, cd)$. Figure 3.2 (a), (b), (c) and (d) show the cases that two loops of $L(ab, cd)$ exist, one of which is loop abc or abd, and the other loop $abcd$ or $abdc$. Figure 3.2(e) shows a case in which four loops in $L(ab, cd)$ exist, loops abc, abd, $abcd$ and $abdc$.

For each of the five cases shown in Figure 3.2, node a runs the algorithm *IfRemove* for each loop. For example, a runs the algorithms *IfRemove(ab, cd, abc)*

(a) Loops abc and $abcd$ (b) Loops abc and $abdc$



(c) Loops abd and $abdc$ (d) Loops abd and $abcd$



(e) Loops $abc, abd, abcd$ and $abdc$

Figure 3.2: Cases of more than one loop of $L(ab, cd)$ existing

and *IfRemove(ab, cd, abcd)* for the case shown in Figure 3.2(a).

If the return value of any of the *IfRemove* algorithms is *True*, ab can be removed due to the loop. In this case, ab can be removed because of cd. Otherwise, ab can not be removed because of cd.

4. If ab can be removed because of cd or any other *2-hop cross links* of it detected by a, node a will remove ab by marking it as *non-routable* and will notify node b about the removal of ab. Otherwise, a will keep ab as *routable*.

Next, we introduce the algorithm *IfRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)*, based on the return value of which node u_1 knows if link u_1u_2 can be removed due to the loop $L(u_1u_2, v_1v_2)$.

The *IfRemove* algorithm takes three parameters as its inputs, where u_1u_2 is the link that is under consideration for removal, v_1v_2 is a *2-hop cross link* of u_1u_2 detected by u_1, and $L(u_1u_2, v_1v_2)$ is a loop that is formed by nodes that u_1u_2 and v_1v_2 are attached to and contains link u_1u_2. The details of the *IfRemove* algorithm are given in Algorithm 3.

If node u_1 receives *AgreeRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* messages from all nodes that it sent the *TryRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* message to, it means there is no other link in the loop $L(ab, cd)$ that is being removed with u_1u_2 concurrently. This prevents the

1. Node u_1 sends a *TryRemove(*u_1u_2, v_1v_2, $L(u_1u_2, v_1v_2)$) message to all other nodes in the loop $L(u_1u_2, v_1v_2)$.

2. When receiving a *TryRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* message from node u_1, node r (i.e., u_2, v_1 or v_2) checks whether any link other than u_1u_2, say link $ru(u = u_1, u_2, v_1$ or $v_2)$, which is contained in the loop $L(u_1u_2, v_1v_2)$ and is attached to it, is trying to be removed due to another loop that contains link u_1u_2.

   This is realized by checking if it sent a *TryRemove(*$ru, xy, L(ru, xy)$*)* message with a higher priority, where $ru \neq u_1u_2$, x and y are any nodes, and $L(ru, xy)$ contains link u_1u_2.

3. If r did not send any such *TryRemove* message(s), it sends an *AgreeRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* message to u_1.

4. If node u_1 receives the *AgreeRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* messages from all nodes that it sent the *TryRemove(*$u_1u_2, v_1v_2, L(u_1u_2, v_1v_2)$*)* message to, the algorithm will return *True*, or otherwise with a value *False*.

Algorithm 3: *The algorithm IfRemove*

removal of u_1u_2 from partitioning the network due to the loop $L(ab, cd)$. In this case, the algorithm will return with the value *True*.

Otherwise, this means another link(s) in the loop $L(ab, cd)$ is being removed with u_1u_2 concurrently. This may cause the partition of the network due to the removal of u_1u_2. In this case, the algorithm will return with the value *False*.

The CLPP graph

The proposed *CLPP* algorithm makes a node keep or remove a link attached to it that has *2-hop cross link(s)* detected by it. The *routable* links generated by the *CLPP* algorithm form a *routable* subgraph of the original network graph, which is called the *CLPP* graph, and is denoted by G_{CLPP}.

Because some links may be removed in the *CLPP* algorithm, graph G_{CLPP} is sparser than the original network graph. However, a node can not detect and remove all kinds of cross-links, so the *CLPP* graph G_{CLPP} is not the final graph for face routing in the proposed *PPCLDP*.

3.3.3 Restricted Cross Link Detection Protocol (RCLDP)

The proposed *Restricted Cross Link Detection Protocol (RCLDP)* uses the *CLPP* graph, G_{CLPP}, generated by the *CLPP* algorithm, as the input graph, and applies the previous work of *CLDP* [32] on graph G_{CLPP}. Therefore, we call it the *Restricted CLDP (RCLDP)*.

In graph G_{CLPP}, many links that have *2-hop cross link(s)* have been removed from the original network graph, so they do not need to be probed using the *Right-Hand Rule* in the *RCLDP*, which have to be probed in the *CLDP* applied on the original network graph [32]. The probe of a link may cause a long face with a large number of edges being traversed, and sometimes several rounds of probing may be needed for a node to make a decision as to whether to keep or remove a link in *CLDP*. Therefore, the communication cost of the *RCLDP* is decreased significantly compared to that of the *CLDP*.

3.3.4 Pre-Processed Cross Link Detection Protocol (PPCLDP)

The proposed *PPCLDP* contains the *2-hop Cross Link Pre-Processing (CLPP)* algorithm and the *Restricted Cross Link Detection Protocol (RCLDP)*.

The *CLPP* algorithm generates a connected subgraph, G_{CLPP}, from the original network graph, in which a node exchanges a few messages with its 1-hop neighbors. The *RCLDP* is a *Restricted CLDP*, which applies the *CLDP* on graph G_{CLPP}, and reduces the number of messages broadcast by nodes significantly compared to the *CLDP* that is applied on the original network graph [32].

The graph generated by *PPCLDP* is called the *PPCLDP* graph and is denoted by G_{PPCLDP}, which is an almost planar subgraph of the original network graph. Next, by proving that G_{CLPP} is a connected graph, we prove that the *GFG* geographic routing never fails on G_{PPCLDP} under realistic environments with obstacles.

Theorem 6 GFG *geographic routing never fails on graph* G_{PPCLDP}.

Proof: For a link, say ab, which is attached to node a, if it is removed by a in the *CLPP*

algorithm, the following three cases must be satisfied.

i) There is a *2-hop cross link* of ab detected by a, say cd.

ii) There must be a loop formed by nodes a, b, c and d that contains link ab, denoted by $L(ab, cd)$.

iii) The return value of the algorithm *IfRemove(*$ab, cd, L(ab, cd)$*)* is *True*. This means that there is no other link in loop $L(ab, cd)$ that is being removed with ab concurrently, based on the algorithm *IfRemove*, shown in Algorithm 3.

In this case, the removal of ab will not partition the network due to the loop $L(ab, cd)$.

For the same reason, the removal of any link in the *CLPP* algorithm will not partition the network. Because the original network is connected, G_{CLPP} is a connected graph.

Because *CLDP* working on a connected graph generates a connected *CLDP*-stable graph [31], *CLDP* working on graph G_{CLPP} generates a connected *CLDP*-stable graph, which is G_{PPCLDP}. Because geographic routing never fails on a connected *CLDP*-stable graph [31], the theorem follows. □

3.4 Simulation Study and Analysis

The proposed *PPCLDP* improves the previous work of *CLDP* with a lower communication cost and faster convergence time. In this section, we confirm the superior performance of *PPCLDP* through a simulation study.

3.4.1 Simulation Settings

In our simulations, the network is a square area of $1000 \times 1000m^2$. Nodes are initially distributed randomly in the area and have a transmission range of 200 meters. The number of nodes in the network is denoted by n. By varying the number n, we change the network density.

We developed our own simulator from scratch to simulate a realistic environment with obstacles. An obstacle between two nodes causes the link between them to break, even

though the *Euclidean* distance between them is less than the transmission range. In the simulations, we use the algorithm *Gen(n, m)* (shown in Algorithm 4) to generate a realistic network graph with n nodes and m obstacles.

1. Randomly distribute n nodes in the network area.
2. Based on nodes' location information, construct a *unit-disk graph, UDG,* where there is an edge between two nodes if and only if the *Euclidean* distance between them is no more than the transmission range.
3. Randomly remove m edges (links) from the *UDG*. The link between two nodes which is randomly removed is called a *broken link.* Note that the *broken links* do not mean the network is partitioned.

Algorithm 4: Algorithm *Gen(n, m)* to generate a realistic network graph

The algorithm *Gen(n, m)* generates a network graph, which is regarded as the original realistic network graph with n nodes and m obstacles, and denoted by G_{orig}. In the simulation study, we simulate two environments, the *less rough* environment and *rough* environment, respectively. In the *less rough* environment, we set the number of obstacles m as $n/2$, or half of the number of nodes in the network. In the *rough* environment, we set the number of obstacles m as n, or the same as the number of nodes in the network.

For each environment, we randomly generate seven network topologies by applying the algorithm *Gen(n, m)*, each of which contains n nodes and m obstacles, where n is $40, 60, 80, 100, 120, 150$ and 200, respectively. For each topology, we ran simulations 30 times with different seeds. Each value shown in the following figures is the average value of those 30 runs.

3.4.2 Example Network Graphs

To give an overview of the network graphs generated by *PPCLDP* and *CLDP*, we show four network graphs in a *rough* environment in Figure 3.3, in which 40 nodes are randomly distributed in the network with 40 obstacles.

Figure 3.3.a, Figure 3.3.b, Figure 3.3.c and Figure 3.3.d, show the original realistic network graph, G_{orig}, the graph generated by *CLDP*, G_{CLDP}, the *CLPP* graph, G_{CLPP}, and the *PPCLDP* graph, G_{PPCLDP}, respectively.

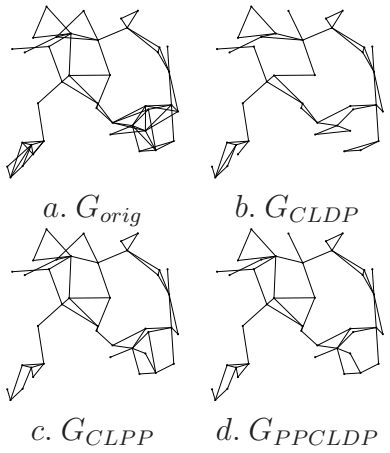


Figure 3.3: An example of network graphs

In Figure 3.3, there is not much difference between graphs G_{CLDP} and G_{PPCLDP}. However, graph G_{CLPP}, which is the input graph of the *RCLDP* of *PPCLDP*, is sparser than graph G_{orig}, which is the input graph of *CLDP* [32]. This is because some cross-links are removed from G_{orig} in the *CLPP* algorithm of *PPCLDP*. Therefore, less probing and traversal of edges are needed in the *RCLDP* of *PPCLDP* than that in the *CLDP* protocol. In addition, very few messages are exchanged in the *CLPP* algorithm of *PPCLDP*. As a result, the communication cost and convergence time of *PPCLDP* are lower than those of *CLDP*, which is corroborated by the simulation results.

3.4.3 Average Number of Cross Link Pairs and 2-Hop Cross Link Pairs

Figure 3.4 and Figure 3.5 show the average number of *cross link pairs* and *2-hop cross link pairs* with respect to the number of nodes, in *less rough* and *rough* environments, respectively. Recall that in *less rough* and *rough* environments, the number of obstacles is half and equal to the number of nodes, respectively.

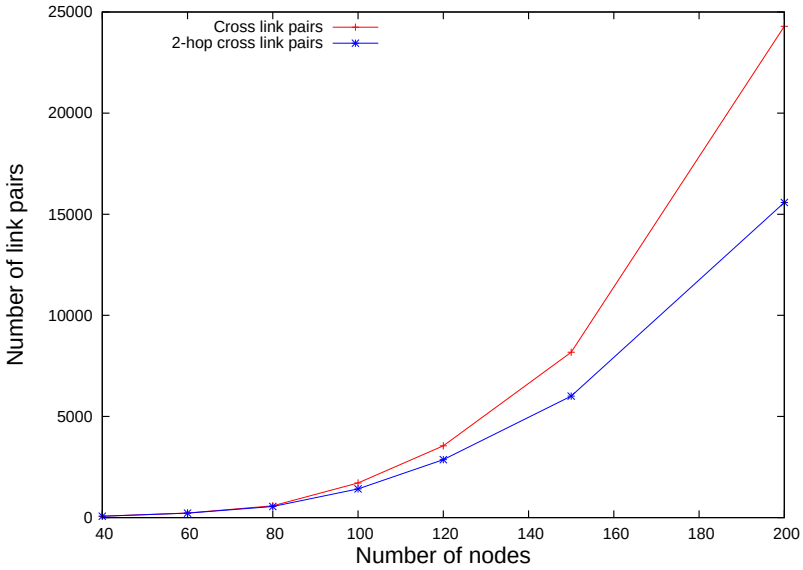


Figure 3.4: Number of *cross link pairs* and *2-hop cross link pairs* in a *less rough* environment

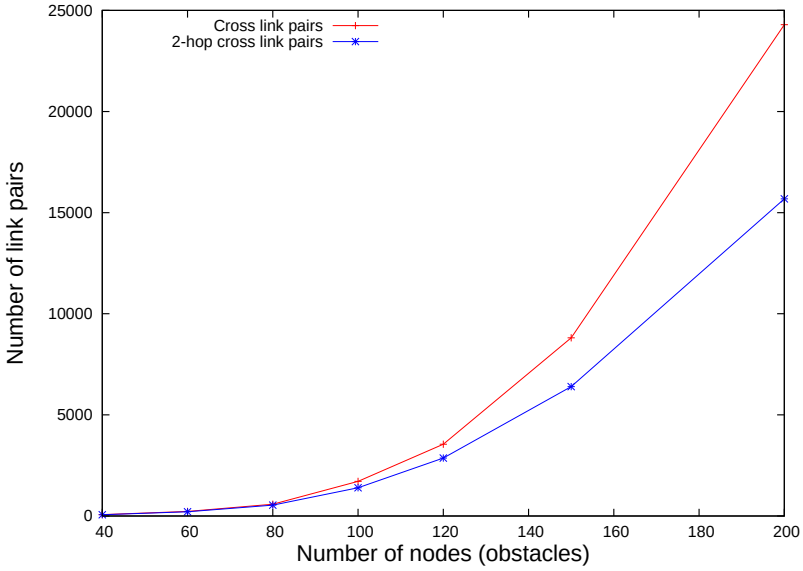


Figure 3.5: Number of *cross link pairs* and *2-hop cross link pairs* in a *rough* environment

We observe that the ratio of the average number of *2-hop cross link pairs* to that of *cross link pairs* is 69% and 82% in *less rough* and *rough* environments, respectively. The proposed *CLPP* algorithm pre-processes the *2-hop cross link pairs* in advance, which gives the *PPCLDP* a better performance than *CLDP*. This will be confirmed by the subsequent simulation results.

3.4.4 Average Number of Edges Removed by *PPCLDP* and *CLDP*

The example graphs in Figure 3.3.b and Figure 3.3.d show that there is not much difference between graphs G_{CLDP} and G_{PPCLDP}. In this section, we show simulation results to confirm this.

Figure 3.6 and Figure 3.7 plot the number of edges removed from the original network graph with respect to the number of nodes, in *less rough* and *rough* environments, respectively.

We observe that the number of edges removed in *PPCLDP* is almost the same as that in *CLDP* in both environments. Therefore, the number of edges kept in graphs G_{PPCLDP} and G_{CLDP} are almost the same in both environments.

3.4.5 Routing Performance

As with *CLDP*, the proposed *PPCLDP* can be used to generate planar graphs for face routing in any existing *GFG* geographic routing protocols.

In our simulations, we use *GPSR* [30] as the underlying *GFG* routing protocol, and apply it on *PPCLDP* and *CLDP*, which are represented by *GPSR/PPCLDP* and *GPSR/CLDP*, respectively. This means we use *PPCLDP* and *CLDP* as the planarization protocols to generate planar graphs for the face routing phase of *GPSR*.

We evaluate the routing performance of *GPSR/PPCLDP* and *GPSR/CLDP* with respect to the metric of the hop count. We randomly select 30% of total nodes as source nodes, and for each source node, we randomly select a destination node. For each pair of source

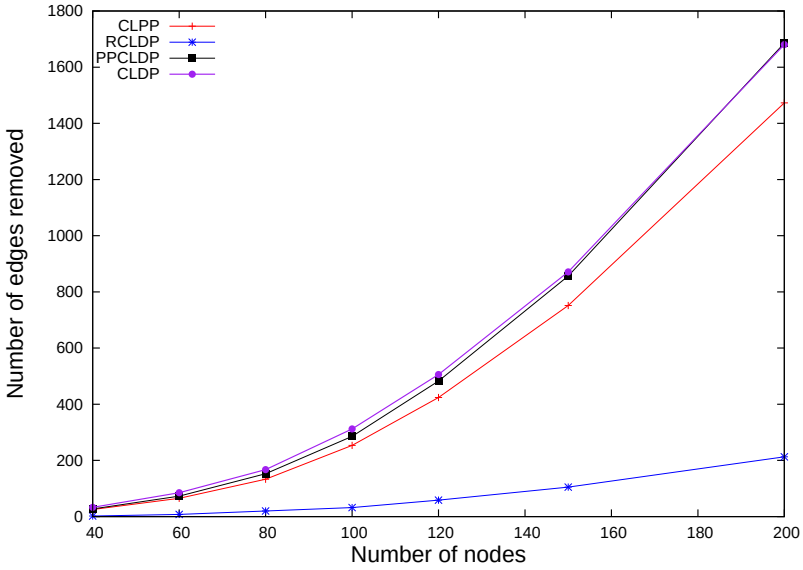


Figure 3.6: Number of edges removed by *PPCLDP* and *CLDP* in a *less rough* environment

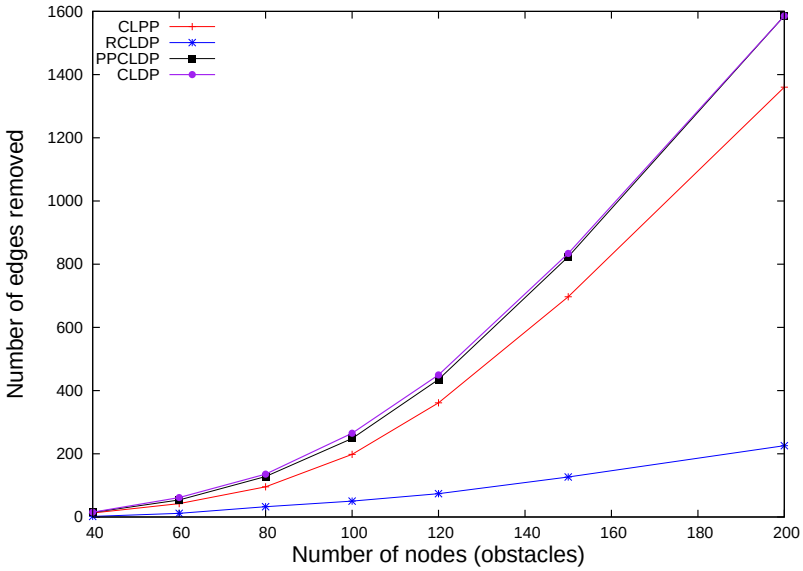


Figure 3.7: Number of edges removed by *PPCLDP* and *CLDP* in a *rough* environment

and destination nodes, we determined how many hops are needed for a packet to be routed from the source to the destination.

Figure 3.8 and Figure 3.9 show the average hop count with respect to the number of nodes, in *less rough* and *rough* environments, respectively. We observe that *GPSR/PPCLDP* and *GPSR/CLDP* perform almost the same in both environments, which is due to the fact that there is not much difference between the planar graphs generated by *PPCLDP* and *CLDP*.

3.4.6 Evaluation of the Communication Costs of PPCLDP and CLDP

We use two metrics, the total number of messages broadcast by all nodes in the network, and the average message size, to evaluate the communication cost of protocols *PPCLDP* and *CLDP*.

Total number of messages sent by all nodes in the network

Figure 3.10 and Figure 3.11 show the total number of messages broadcast by all nodes with respect to the number of nodes, in *less rough* and *rough* environments, respectively.

We observe that the total number of messages sent in the *CLPP* algorithm of *PPCLDP* is very low, 97% less than that in *CLDP* protocol in both environments. The total number of messages broadcast by nodes in the *RCLDP* of *PPCLDP* is 68% and 61% less than that in the *CLDP* in the *less rough* and *rough* environments, respectively.

Average message size

The average message size is measured by the sum of the size of all messages sent by nodes in the network divided by the number of messages.

We plot the average message size in bytes with respect to the number of nodes, in *less rough* and *rough* environments in Figure 3.12 and Figure 3.13, respectively. The average message size in *CLPP* algorithm is 28 bytes, which is much smaller than that in the *CLDP* protocol in both environments. The *RCLDP* also performs well in the average message size

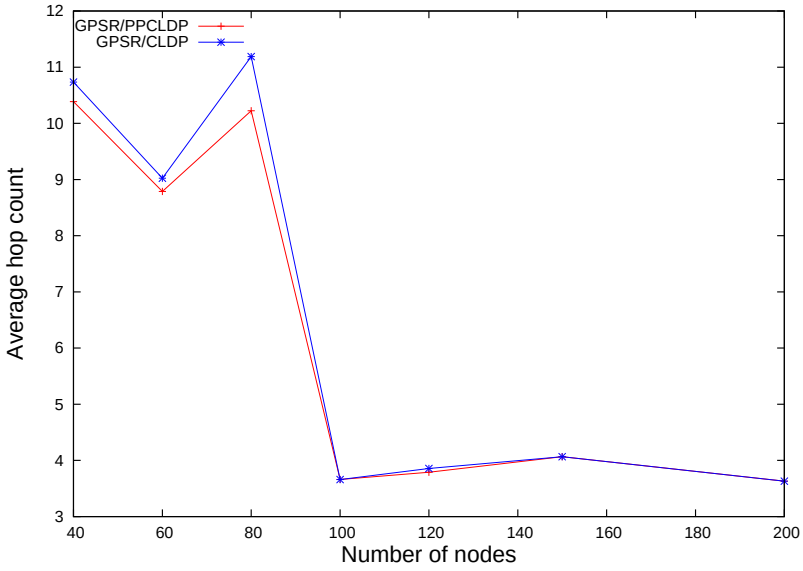


Figure 3.8: Average hop count for *GPSR/PPCLDP* and *GPSR/CLDP* in a *less rough* environment

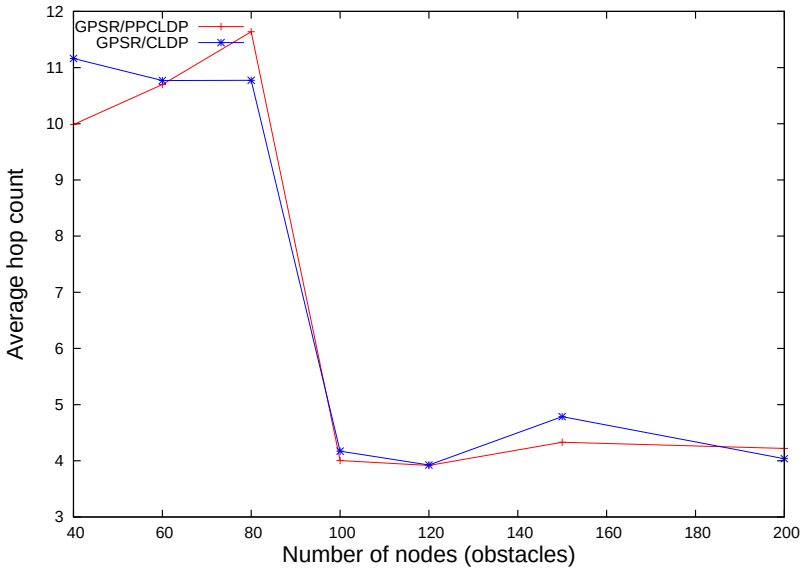


Figure 3.9: Average hop count for *GPSR/PPCLDP* and *GPSR/CLDP* in a *rough* environment

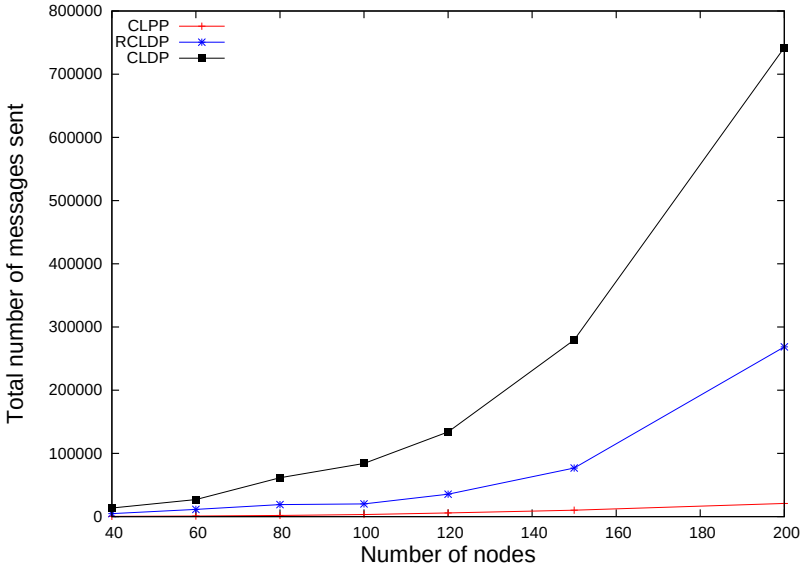


Figure 3.10: Total number of messages sent by all nodes in a *less rough* environment

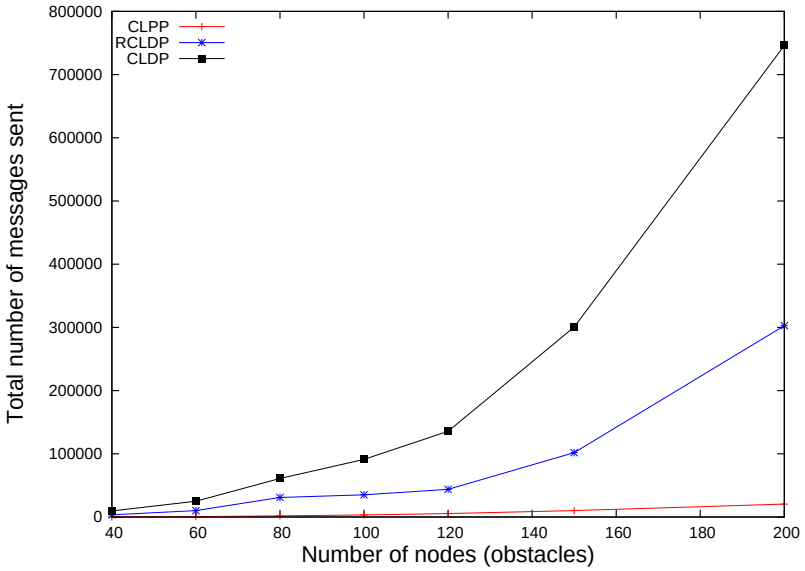


Figure 3.11: Total number of messages sent by all nodes in a *rough* environment

than the *CLDP* protocol in both environments.

Communication Costs of PPCLDP and CLDP

We present the communication costs in bytes with respect to the number of nodes of *PP-CLDP* and *CLDP* in *less rough* and *rough* environments in Figure 3.14 and Figure 3.15, respectively. Because the proposed *PPCLDP* contains the *CLPP* algorithm and the *RCLDP*, we computed the communication cost of *PPCLDP* as the sum of both. The average communication cost in bytes of *PPCLDP* is 72% and 65% less than that of *CLDP* in *less rough* and *rough* environments, respectively. This confirms that the proposed *PPCLDP* performs better than *CLDP* with respect to communication cost.

3.4.7 Evaluation of the Average Convergence Time of PPCLDP and CLDP

In this section, we use the convergence time of protocols *PPCLDP* and *CLDP* as another metric to compare performance.

We assume the one-hop message transmission latency is T: it takes an average time T for a message to travel from a node to its neighbors. We define the convergence time of protocols *PPCLDP* and *CLDP* as the time interval, in terms of T, from the time the first node starts sending messages to the time that no more nodes need to send messages to probe edges in each protocol.

We depict the convergence time in T with respect to the number of nodes in *less rough* and *rough* environments in Figure 3.16 and Figure 3.17, respectively. We observe that the average convergence time of *PPCLDP* is 29% and 45% less than that of *CLDP* in *less rough* and *rough* environments, respectively. Note that the convergence time of *PPCLDP* is the sum of that of both the *CLPP* and the *RCLDP* algorithms. This also confirms that the proposed *PPCLDP* outperforms *CLDP* in convergence time.

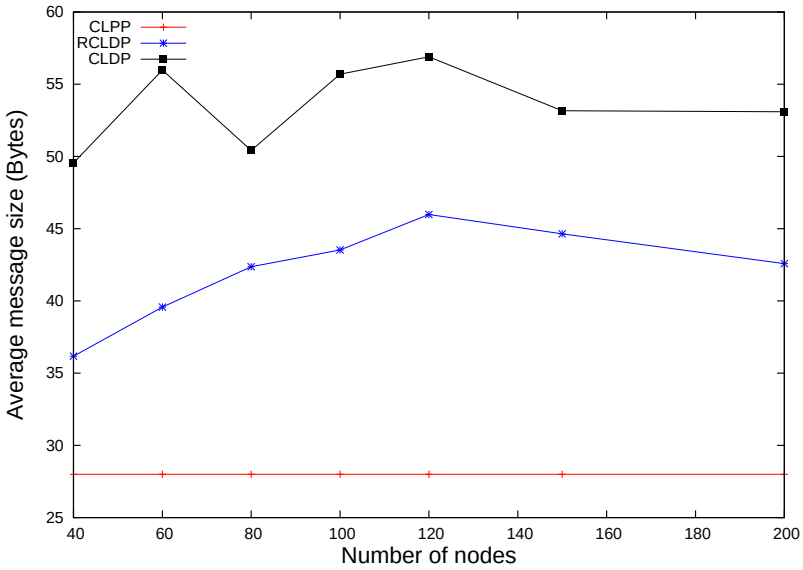


Figure 3.12: Average size of messages sent in *PPCLDP* and *CLDP* in a *less rough* environment

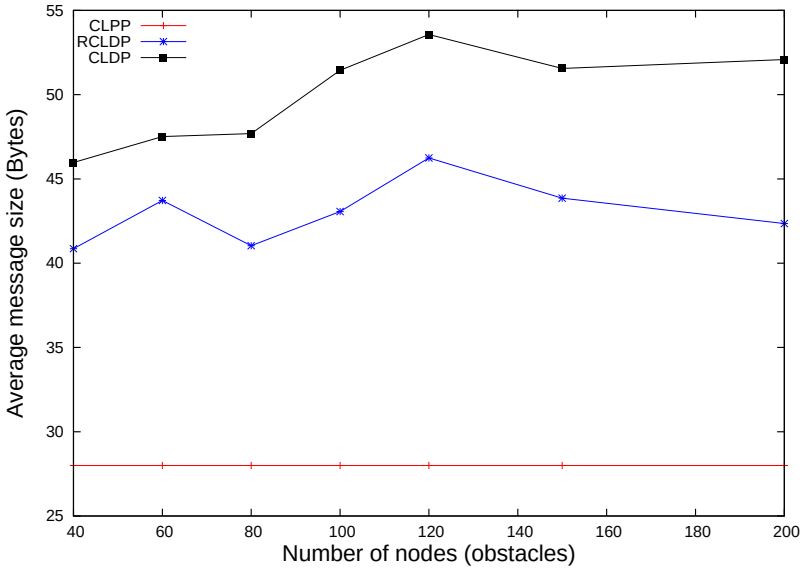


Figure 3.13: Average size of messages sent in *PPCLDP* and *CLDP* in a *rough* environment

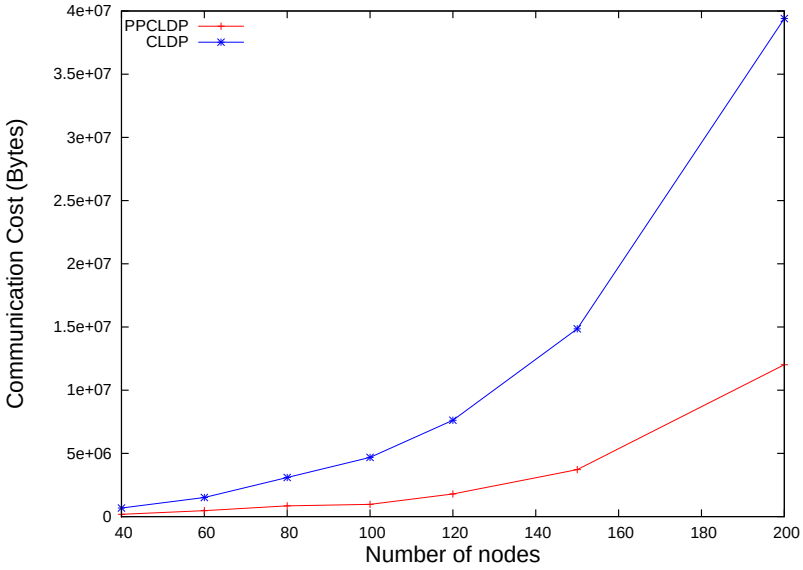


Figure 3.14: Communication cost of *PPCLDP* and *CLDP* in a *less rough* environment

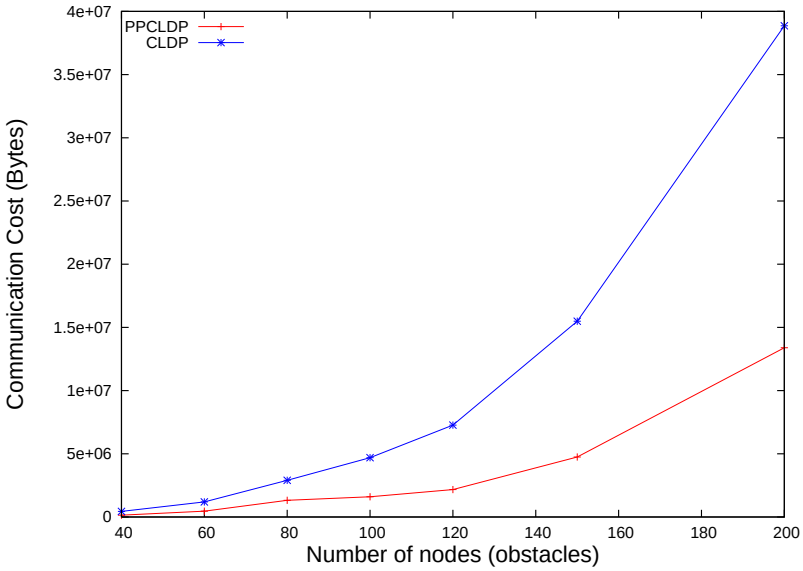


Figure 3.15: Communication cost of *PPCLDP* and *CLDP* in a *rough* environment

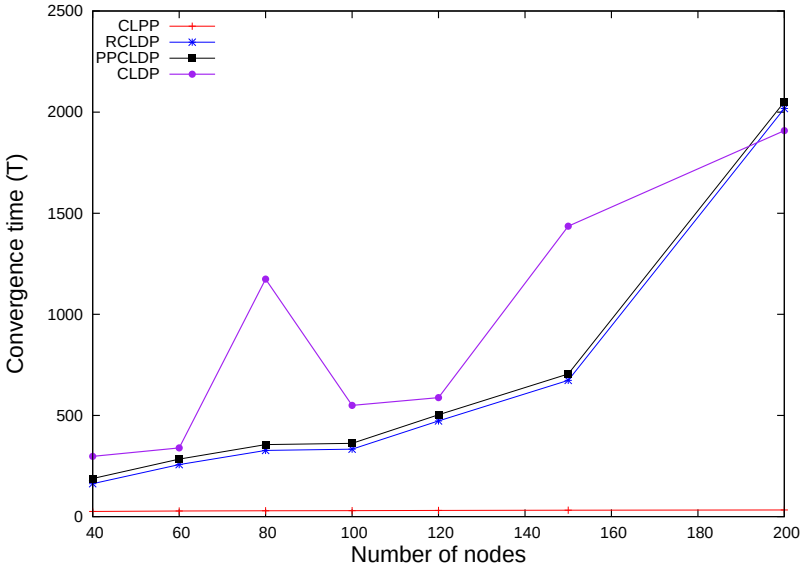


Figure 3.16: Convergence time of *PPCLDP* and *CLDP* in a *less rough* environment

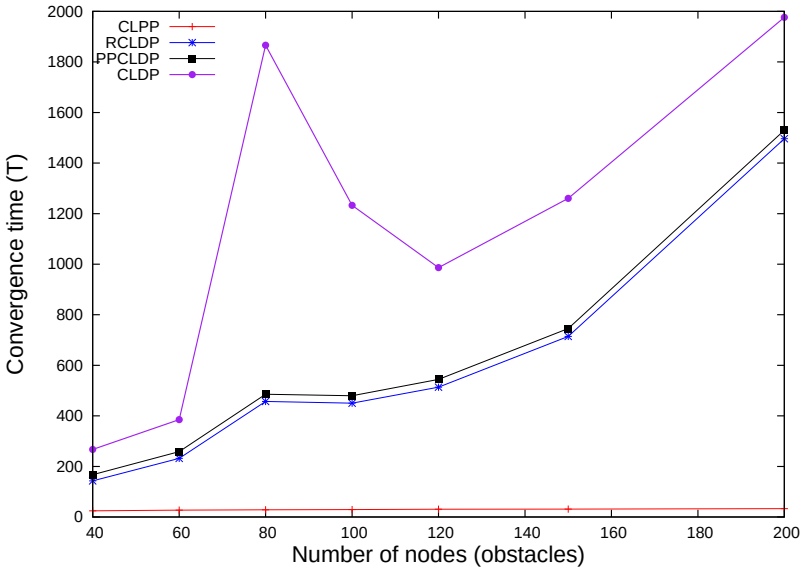


Figure 3.17: Convergence time of *PPCLDP* and *CLDP* in a *rough* environment

3.5 Related Work

Kim et al. [33] describe ways in which the existing planarization algorithms may fail in realistic environments. They also demonstrate the consequences of geographic routing in a real wireless testbed.

Seada et al. [56] describe that many state-of-the-art techniques (like *GPS*) usually cause location errors (errors in location information). They also analyze the effect of location errors on the accuracy and performance of geographic routing in sensor networks. They introduce a local fix for face routing to deal with the location errors in the network. Their scheme guarantees network connectivity, but may keep some cross-links that should be removed in the planarization phase.

Yun et al. [66] propose a localization algorithm, which predicts the positions of sensor nodes based on a hop progress analytical model for a given network topology in WSNs.

Barriere et al. [6] propose a variant of a face routing protocol for the ad hoc networks with obstacles, which guarantees message delivery if the ratio of maximum and minimum transmission range is at most $\sqrt{2}$.

Ansari et al. [3] propose a generalization of face routing, called FRONC, which guarantees delivery in a graph with disjoint crossing edges.

Chavez et al. [13] give an algorithm to construct a spanner of a *unit-disk graph* with nodes of irregular transmission ranges. They also show that the spanner is planar if the distance between any two nodes is at least $\sqrt{1-r^2}$, where r is a parameter.

Kim et al. [32] propose a *Cross-Link Detection protocol, CLDP*, which makes geographic routing work correctly on arbitrary connectivity graphs.

Ben et al. [37] propose a geographic routing protocol, *GDSTR*, which uses greedy routing first, tree routing when greedy fails, and greedy routing again when it is possible to do so. In the tree routing, packets are routed on hull trees instead of planar faces as in face routing. Therefore, the planarization of graphs required when greedy routing fails in most of the existing *GFG* geographic routing protocols, is not needed in *GDSTR*. In some cases,

the *GDSTR* may use a much longer route than existing *GFG* routing protocols, however. The reasons are (i) the destination may not be a descendant of a node even though it is in the convex hull of the node in *GDSTR*, and (ii) hull trees in *GDSTR* may not approximate voids as well as planar faces in existing *GFG* routing protocols. In addition, in *GDSTR*, any packets with source and destination nodes in subtrees rooted at different children of the root of a hull tree will be definitely routed through the root. This may cause a high communication cost at the root, which is not desirable for mobile ad hoc and sensor networks.

3.6 Chapter Summary

In mobile ad hoc and sensor networks, most geographic routing protocols, e.g., *GFG* routing protocols, make an ideal assumption that the network graph is a *UDG*, on which existing planarization algorithms are applied for face routing. However, in realistic environments, the assumption of *UDG* may be violated, which may cause the current *GFG* geographic routing not to work correctly.

In this chapter, we proposed a *Pre-Processed Cross Link Detection Protocol, PPCLDP*, which extracts an almost planar graph, G_{PPCLDP}, from a network graph under realistic environment with obstacles. We proved that *GFG* geographic routing never fails on graph G_{PPCLDP}.

The proposed *PPCLDP* contains a *2-hop Cross Link Pre-Processing (CLPP)* algorithm and a *Restricted Cross Link Detection Protocol (RCLDP)*. In the *CLPP* algorithm, a node detects any *2-hop cross links* of a link attached to it and decides whether to keep or remove the link by exchanging a few messages with its neighbors. The *CLPP* algorithm generates a graph, G_{CLPP}, which is the input graph of the *RCLDP* and is sparser than the input graph of the *CLDP*. This results in much fewer probing messages needed in the *RCLDP* than in *CLDP*. The significantly reduced number of broadcast messages causes *PPCLDP* to have a much lower communication cost and faster convergence time than *CLDP*. Our simulation results show that the average communication cost and convergence time of *PPCLDP* are,

respectively, 65% and 45% lower than those of *CLDP*. This confirms that the proposed *PPCLDP* is more suitable for mobile ad hoc and sensor networks than *CLDP*.

Chapter 4

A Hill-Area-Restricted (HAR) Geographic Routing Protocol for MANET

4.1 Introduction

In mobile ad hoc and sensor networks, a variety of geographic routing schemes [10, 30, 34, 61] have been developed recently, and most belong to the Greedy-Face-Greedy (GFG) routing category. Compared to topology-based routing schemes, GFG routing schemes need each node to maintain the position information of neighbors, instead of a large routing table, which is more desirable in mobile ad hoc and sensor networks.

In most existing GFG geographic routing protocols, like *GPSR* [30], greedy routing is applied first, which may fail at a concave node, a node closer to the destination than any of its neighbors [30]. The reason for this is that in geographic routing protocols, only a neighbor's position is known to each node, hence, a node can not make routing decisions based on the whole network topology. Therefore, nodes may not select next hop neighbors wisely in some networks with special topologies, e.g., networks with voids or obstacles. This may cause packets to enter concave areas and reach concave nodes, and cause greedy routing to fail. Face routing is applied to recover from greedy routing failures. which may cause many extra hops in routing and decrease routing efficiency.

Figure 4.1 shows an example network topology, where there is a lake that contains no

nodes in the network. Nodes are distributed on the ground area that contains a peninsula in the shaded area. Suppose nodes s and d are the source and the destination, respectively. Greedy routing causes the packet from s to enter the peninsula and reach a concave node c very deep in the peninsula, which fails to find a neighbor closer to the destination than itself and causes greedy routing to fail. In this case, face routing is applied, which helps the packet come out of the peninsula through node e. The route of entering and retreating the peninsula is shown as a red line in Figure 4.1, which decreases the routing efficiency in terms of hop count and delay.

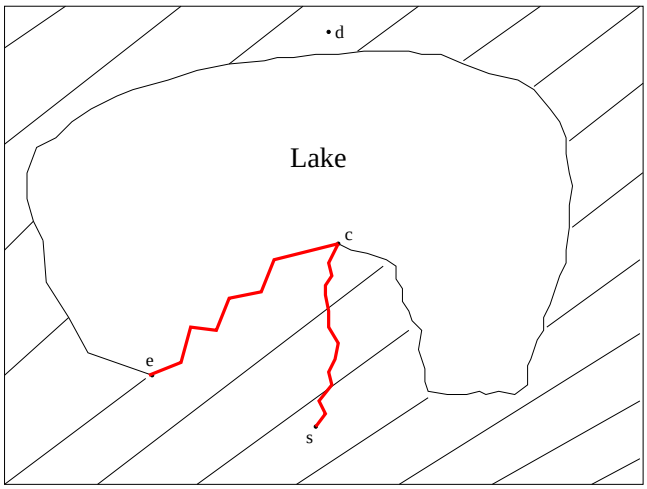


Figure 4.1: Example of a route entering and retreating a peninsula

4.1.1 Research Motivations

Noa et al. [4] propose a *Node Elevation Ad hoc Routing (NEAR)* protocol to improve efficiency. It consists of three algorithms, a node reposition algorithm, a void bypass algorithm and a routing algorithm, and improves the overall efficiency of greedy and perimeter routing of the original GFG routing. However, there are two problems with *NEAR*.

First, the Right-Hand Rule (RHR) is used on a non-planar graph in the void bypass algorithm, which fails to find paths around voids in some cases. As a result, packet delivery is not guaranteed in the network, which is not desirable in mobile ad hoc and sensor

networks. Second, the void bypass algorithm in *NEAR* incurs a high communication cost due to control messages.

This motivated our work in this chapter. We develop a geographic routing protocol, which improves the overall greedy and face routing of GFG routing, guarantees the packet delivery in the network, and lowers the communication cost.

4.1.2 Main Contributions

In this chapter, we propose a Hill-Area-Restricted (HAR) routing protocol, which contains a Concave Area Identification (CAI) algorithm, a Removing Hill Area (RHA) algorithm, and a Hill-Area-Restricted GPSR (HAR-GPSR) routing algorithm.

The CAI algorithm identifies concave nodes and concave areas in advance. The RHA algorithm removes concave areas, which prevents packets from entering the concave areas that the destination nodes do not reside in. The communication cost of the CAI and RHA algorithm in HAR is lowered simultaneously compared to that of the first two algorithms in NEAR.

The HAR-GPSR routing algorithm is based on the GPSR [30], and uses the results of the first two algorithms as its inputs. Compared to the previous work of GPSR, the proposed HAR is more efficient in terms of hop count. Compared to the previous work of the NEAR, the HAR-GPSR guarantees packet delivery in the network.

In the rest of this chapter, Section 4.2 presents preliminaries, Section 4.3 describes the proposed HAR and Section 4.4 presents the simulation results on the HAR and the NEAR. Section 4.5 describes the related work. Section 4.6 concludes the chapter.

4.2 Preliminaries

In this section, we discuss the previous work of *NEAR* [4]. The *NEAR* protocol contains three algorithms, the node reposition algorithm, the void bypass algorithm, and the routing algorithm. Each algorithm runs distributively, and the results of the first two algorithms

serve as input for the third one.

4.2.1 The Node Reposition Algorithm

At the network start-up, each node runs the node reposition algorithm distributively, which identifies and marks concave nodes. In their algorithm, the concave nodes are those with wide sense concavity. They define that a node has wide sense concavity if some destination can not be reached through any of its neighbors using only the greedy process.

Suppose nodes have two dimensional real coordinates, (x, y). Each node is assigned a virtual coordinate, (x, y, z), in which x and y are set to the same value as those of its real coordinate, and z as its height is set to 0 initially.

If it has at least two neighbor nodes with 0 height, a node checks if the angle between any two adjacent neighbors with 0 height exceeds a threshold angle, α, which is usually larger than π. If so, it is identified as a wide sense concave node and sets the virtual third dimension coordinate, z, to 1. Its new virtual (x, y) position will be the average position of the two neighbors. In this case, the node is elevated and is called a floating node. Otherwise, if the angle between any two adjacent neighbors with 0 height of it does not exceed α, the node's virtual coordinate will be kept unchanged.

If a node has less than two neighbors with height 0, then its virtual (x, y) position will be the average (x, y) position of all neighbors with minimal height, and its virtual coordinate, z, will be one above that.

The result of the reposition algorithm is that each concave node is elevated by assigning the z coordinate with a value greater than 0. An elevated node is called a floating node.

4.2.2 The Void Bypass Algorithm

The void bypass algorithm is run after the node reposition algorithm, which aims to find paths around voids a priori.

For a non-floating node, say u, if the angle between any two adjacent neighbors with 0 height of it, say v and w, is greater than a threshold angle, β $(\pi \leq \beta < \alpha)$, which is usually

π, then it generates a void discovery message with a randomly selected identification and sends it to one of the two neighbors, v or w. Suppose u sends the message to the counter-clockwise neighbor, say v. Node v will forward the message to a neighbor node using the right-hand rule (RHR) [8]. If the message comes back to u from node w, then u keeps w and v as the clockwise and counterclockwise next hop neighbors of the void, respectively.

4.2.3 Routing Algorithm

The packet that needs to be routed in the network contains six parameters: the virtual and real coordinates of the destination; the current routing mode, which is greedy or perimeter; the starting point of current perimeter routing; the current void bypass direction (clockwise or counterclockwise); and the identification of the current void bypassed.

The routing algorithm first checks for a special case, which is if the packet reaches the very close vicinity of the destination, by checking if the virtual and physical coordinates of the current node are close enough to those of the destination node. For example, if the virtual coordinates are within the transmission range, and if the physical coordinates are within the destination node's height times the transmission range.

If so, the allowed maximum virtual Z coordinate, z_{max}, is set as the destination node's Z coordinate, z_{dest}. Otherwise, if the current node's Z coordinate, z_{curr}, is greater than 0, then z_{max} is assigned as $z_{curr} - 1$. In other cases, z_{max} is set as 0.

The routing mode of a packet is initially set to greedy at the source node. When the packet is in greedy mode, the next node with the Z coordinate less than or equal to z_{max} will be selected by greedy routing. If the greedy fails, perimeter routing will be applied, which uses the predefined void bypass routes. The packet will enter greedy routing mode again when possible.

4.2.4 Problems Existing in NEAR

The *NEAR* protocol improves both the greedy routing and the perimeter routing. However, it has two problems.

The first is in the void bypass algorithm, where the void discovery message is forwarded using the RHR on a non-planar graph. However, the RHR is known to guarantee traversing around the boundary of a closed polygon only in a planar graph [8]. Though the forwarding using RHR on a non-planar graph may succeed 99.5% of the times [28], it is not desirable for geographic routing in mobile ad hoc and sensor networks.

The second problem is also in the void bypass algorithm, where nodes need to transfer a large number of messages to discover the voids they reside in. This results in high communication cost, which is not desirable in mobile ad hoc and sensor networks.

4.3 Hill-Area-Restricted (HAR) Geographic Routing Protocol

In this section, we propose a Hill-Area-Restricted (HAR) geographic routing protocol, which contains three algorithms, the Concave Area Identification (CAI) algorithm, the Removing Hill Area (RHA) algorithm and the Hill-Area-Restricted GPSR (HAR-GPSR) routing algorithm. Before presenting the HAR, we first introduce the assumptions.

4.3.1 Assumptions

1. By *GPS* or other positioning services [12, 23], each node knows its own location information.

2. Each node knows all its 1-hop neighbors' location information through *Hello* messages.

3. All nodes have the same transmission range.

4. Nodes in the network construct a *unit-disk graph, UDG*, where there is an edge between two nodes if and only if the Euclidean distance between them is at most the transmission range.

5. The network is connected.

4.3.2 Concave Area Identification (CAI) Algorithm

The Concave Area Identification (CAI) algorithm identifies each concave area, which consists of connected concave nodes, with a unique identification. Based on this and the position of a destination node, nodes can make decisions regarding whether to enter a concave area or not when routing a packet.

The Concave Area Identification (CAI) algorithm is run at the network start-up by each node distributively. It identifies a concave node and elevates it by assigning it an additional dimension coordinate with a value greater than 0. It also identifies a concave area and assigns it a unique id.

Virtual coordinate

Suppose a node, say u, in the network has a real two dimension coordinate, denoted by (u_x, u_y). Based on this, node u generates a virtual coordinate, denoted by (u_x, u_y, u_z), where u_x and u_y are the same as those in its real coordinate, and u_z is the virtual third dimension coordinate, which is set to 0 initially. Note that u_z is regarded as the height of u. Each node in the network piggybacks its virtual coordinate in a Hello message, which is broadcast periodically. This is how a node knows the virtual coordinates of all its neighbors.

Concave nodes

Before introducing concave nodes, a few definitions are given below:

Definition 11 *The neighbors* [1] *with 0 height of node* u *are called the Ground Neighbors of* u.

Definition 12 *The set that contains all the Ground Neighbors of* u *is called the Ground Neighborhood Set of* u*, and is denoted by* $GN(u)$.

[1] In the rest of this chapter, a node's neighbors refers to the 1-hop neighbors of the node.

Definition 13 *The angle of a ground neighbor of u, say v, denoted by* $\angle v_u$*, which belongs to* $[0, 2\pi]$*, is defined as* $\arctan(v_y - u_y)/(v_x - u_x)$ *when* $\angle v_u \in [0, \pi/2]$*,* $\pi + \arctan(v_y - u_y)/(v_x - u_x)$ *when* $\angle v_u \in (\pi/2, 3\pi/2]$*, and* $2\pi + \arctan(v_y - u_y)/(v_x - u_x)$ *when* $\angle v_u \in (3\pi/2, 2\pi]$*, where* (u_x, u_y) *and* (v_x, v_y) *are the real coordinates of nodes* u *and* v*, respectively.*

Definition 14 *Based on an increasing order of the angles of all ground neighbors of node u, all ground neighbors of u are inserted into an array in the order* [2]*. The array is called the Ordered Ground Neighborhood Array of node* u*, and is denoted by* $Ord(GN(u))$*.*

Definition 15 *Suppose there are* $m(m \geq 0)$ *nodes in Ord(GN(u)). The Maximum Angle between* u*'s Ground Neighbors, denoted by* $Maxangle(GN(u))$*, is defined as* 2π *when* $m = 0$ *or* 1*, and is defined as* $max(|\angle w_u - \angle v_u|, 2\pi - |\angle l_u - \angle f_u|)$*, where v and w are any two adjacent nodes in Ord(GN(u)) and f and l are the first and last nodes in Ord(GN(u)), when* $m \geq 2$*.*

Definition 16 *For a node* u*, if it satisfies that* $Maxangle(GN(u)) > \alpha$*, where* α *is a threshold angle that is greater than* π*, then* u *is called a Concave Node.*

A *Concave Node* in our proposed HAR routing protocol is defined based on the *Maximum Angle between a node's Ground Neighbors*, which is slightly different from that in the NEAR [4] protocol.

Concave area

Having defined concave nodes, the definition of a concave area is included as:

Definition 17 *A concave area is defined as a subgraph of the original network graph, which is denoted by* $CA(N_{CA}, E_{CA})$*, where* N_{CA} *is the node set that contains all concave*

[2] For two ground neighbors of node u, say v and w, if $\angle v_u = \angle w_u$, then they can be inserted into the array in either order.

nodes that are connected with each other directly or through other concave nodes in N_{CA}, and E_{CA} is the edge set that contains all edges between nodes in N_{CA}.

Observation 1 *A concave area is a connected graph.*

Observation 2 *There is no edge between any two concave nodes that are in different concave areas.*

A concave area is assigned a unique id, which is called the identification of the concave area and is denoted by CId.

CAI algorithm

A node, say u, has a real coordinate (u_x, u_y) and a virtual coordinate (u_x, u_y, u_z), where u_z is set to 0 initially. Node u keeps a variable CId, which denotes the identification of the concave area that it may reside in and is set as -1 initially. Let $N(u)$ denote the set that contains all neighbors of u. When receiving a Hello message from one of its neighbors, node u runs the *CAI* algorithm as shown in Algorithm 5.

if $Maxangle(GN(u)) > \alpha$ **then**
  if $\{v_z = 0 \mid \forall v \in N(u) \cup \{u\}\}$ **then**
    $CId_u = Id_u$;
  else
    $CId_u = max\{CId_v | \forall v \in N(u) \cup \{u\}\}$;
  end if
  $u_z = min\{v_z \mid \forall v \in N(u)\} + 1$;
else
  if $u_z > 0$ **then**
    $u_z = 0$;
    $CId_u = -1$;
  else
    Return;
  end if
end if
send a $Hello((u_x, u_y, u_z), CId_u)$ message;

Algorithm 5: *The CAI algorithm*

Node u first checks if it is a concave node by checking if $Maxangle(GN(u)) > \alpha$, where α is a threshold angle that is greater than π. If so, it will update the identification of the concave area CId_u it resides in as follows:

1. If u and all its neighbors are at height 0, it updates the CId_u field to its identification Id_u, which is treated as a potential identification of the concave area that it resides in. u is regarded as a potential initiator of the concave area[3].

2. Otherwise, it updates CId_u to the largest of the CIds of all its neighbors and itself. Thus, only one CId is kept by each concave node in a concave area.

In addition, u will be elevated by setting u_z as one above the minimum height of all its neighbors.

If node u finds that $Maxangle(GN(u)) \leq \alpha$, which means it is not a Concave Node, it further checks if $u_z > 0$.

If so, which means it was a concave node previously but is not now, it sets u_z as 0 and CId_u as -1. Otherwise, it returns from the algorithm. This makes a node that is not a Concave Node, say u, stay on the ground by setting u_z as 0 and CId_u as -1. Note that the u_x and u_y are never changed and are the same as those in its real coordinate.

Node u piggybacks its virtual coordinate (u_x, u_y, u_z) and CId_u in a Hello message, which is broadcast to all its neighbors. By all nodes broadcasting the Hello messages, each node knows the virtual coordinates and the values of the variable CId of all its neighbors.

Results of the CAI algorithm

Suppose a node u sets its virtual coordinate as $(u_x, u_y, 0)$ and the identification of the concave area that it may reside in, CId_u, as -1 initially. The CAI algorithm makes u keep or update u_z and CId_u as follows:

[3] Sometimes, several nodes may initiate different CIds for the same Concave Area.

1. If u is a concave node, it updates its virtual coordinate to (u_x, u_y, u_z), where $u_z \geq 1$. CId_u is also updated to a new value, say Id_i, where $Id_i \geq 0$.

2. If u is not a concave node, it will keep its virtual coordinate and CId_u as $(u_x, u_y, 0)$ and -1, respectively.

Figure 4.2 shows an example of the result of the CAI algorithm, where there are five nodes in the network, q, p, w, v, and u, the real two dimension coordinates of which are $q(0,0)$, $p(0,1)$, $w(1,0)$, $v(1,1)$ and $u(1.5,1.5)$, respectively. Based on this, they initially generate their virtual coordinates as $q(0,0,0)$, $p(0,1,0)$, $w(1,0,0)$, $v(1,1,0)$ and $u(1.5,1.5,0)$, respectively. The identification of the concave area that they may reside in, which are the values of variable CId, are all set to -1 initially.

Suppose u is a Concave Node, whose id is 5, while the other four nodes are not. The result of the CAI algorithm is that u is virtually elevated to the position of u' as shown in Figure 4.2 by updating its virtual coordinate to $(1.5,1.5,1)$, and CId_u is updated to 5. The virtual coordinates and the value of variable CId of all the other four nodes do not change.

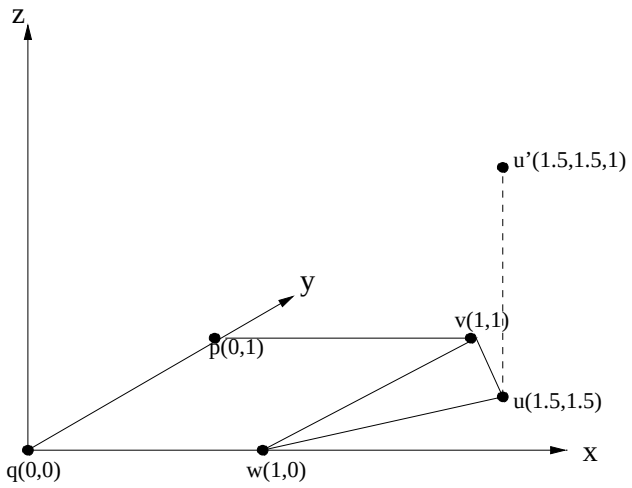


Figure 4.2: Example of the result of CAI

Node types

After executing the CAI algorithm, each node is defined as a ground, foothill (i.e., bump or gate), or onhill node as its node type, based on its virtual z coordinate, as follows:

Definition 18 *Node u is called a ground node if* $u_z = 0$.

Definition 19 *Node u is called a foothill node if* $u_z = 1$.

Definition 20 *Node u is called an Onhill node if* $u_z \geq 2$.

Definition 21 *The foothill nodes that have onhill nodes with height* 2 *as its neighbors are called gate nodes.*

Definition 22 *The foothill nodes that are not gate nodes are called bump nodes.*

Observation 3 *A concave node* u *with* $u_z \geq 1$ *may be a foothill (i.e., bump or gate) or an onhill node.*

Lemma 11 *If the CAI algorithm makes a concave node* u *with height* u_z*, then at least one of the neighbors of* u *has height* $u_z - 1$.

Proof: Based on the CAI algorithm shown in Algorithm 5, u_z is set as one above the minimum height among all its neighbors. Therefore, there must be a neighbor v of u that satisfies that $v_z + 1 = u_z$, which means $v_z = u_z - 1$. The Lemma follows. □

Lemma 12 *The* z *coordinate* v_z *of a neighbor node* v *of a concave node* u *must satisfy* $u_z - 1 \leq v_z \leq u_z + 1$.

Proof: Suppose there is a neighbor node w of node u, which satisfies that $w_z > u_z + 1$ or $w_z < u_z - 1$.

If $w_z > u_z + 1$, then $w_z > u_z$. Based on the CAI algorithm shown in Algorithm 5, node w sets w_z as one above the minimum height of all its neighbors, which include node

u. Let node m be a neighbor of w, which has the minimum height of all w's neighbors. Then $m_z \leq u_z$ and $w_z = m_z + 1$, which makes $w_z \leq u_z + 1$. This contradicts the fact that $w_z > u_z + 1$.

If $w_z < u_z - 1$, then $w_z < u_z$. Based on the CAI algorithm shown in Algorithm 5, node u sets u_z as one above the minimum height of all its neighbors, which include node w. Let node m be a neighbor of u, which has the minimum height of all u's neighbors. Then $m_z \leq w_z$ and $u_z = m_z + 1$, which makes $w_z \geq u_z - 1$. This contradicts the fact that $w_z < u_z - 1$. Thus, the Lemma follows. □

Corollary 5 *A gate node must have at least one ground node and one onhill node with height 2 as its neighbor.*

Corollary 6 *An onhill node with height 2 must have at least one gate node, and can not have ground or bump nodes as its neighbors.*

Corollary 7 *An onhill node with height greater than 2, can not have ground or foothill (i.e., bump or gate) nodes as its neighbors.*

Lemma 13 *An onhill node must reach a ground or bump node through at least one gate node.*

Proof: Based on the assumption that the original network is connected, an onhill node can reach each ground or bump node.

Suppose there is an onhill node o that can reach a ground or bump node, say u, not through any gate node. Let R be the route from node o to u, which contains $m(m \geq 0)$ onhill nodes except o, denoted by $o_1, o_2, \cdots, o_m(m \geq 0)$, where o_m is the last onhill node in route R and is node o itself when $m = 0$.

We divide the route R into two subroutes, which are the route from o to o_m, denoted by R_1, and the route from o_m to u, denoted by R_2, as shown in Figure 4.3 below. Note that

there are no other onhill nodes except o_m in R_2, which means nodes in R_2 except o_m are all ground or foothill nodes.

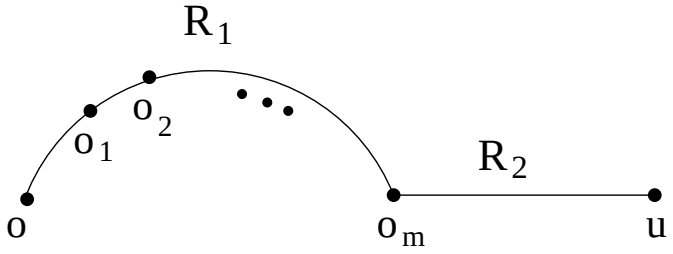


Figure 4.3: Routes from o to u

From Corollary 7, the height of o_m must be 2. From Corollary 6, there must be a gate node connected to o_m directly in route R_2. This contradicts the fact that node o can reach node u without any gate node on the path. Thus, the Lemma follows. □

Lemma 14 *A ground or bump node can be reached by at least one gate node directly or through other ground, bump or gate nodes (i.e., not through onhill nodes).*

Proof: Based on the assumption that the original network is connected, a ground or bump node can be reached by any gate node.

Suppose all gate nodes must reach a ground or bump node, say u, through onhill node(s). Let node g be a gate node, which reaches u through onhill nodes $o_1, o_2, \cdots, o_m (m \geq 1)$, where o_m is the last onhill node in the route from g to u, denoted by R.

We divide the route R into two subroutes, which are the route from g to o_m, denoted by R_1, and the route from o_m to u, denoted by R_2, as shown in Figure 4.4. Note that there are no other onhill nodes in R_2 except o_m.

From Lemma 13, there must be a gate node, say g', in route R_2. This means node g' can reach node u not through onhill nodes, which contradicts the fact that all gate nodes must reach a ground or bump node through onhill nodes. Thus, the Lemma follows. □

Corollary 8 *All ground and bump nodes can be reached by gate nodes directly or through nodes that are not onhill nodes.*

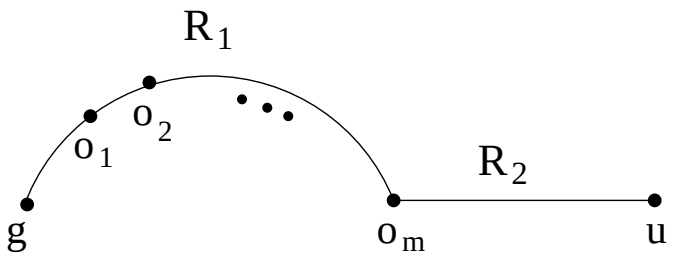


Figure 4.4: Routes from g to u

Hill area

With the definition of concave area and node types, we introduce the definition of a hill area as follows:

Definition 23 *The maximum height of concave nodes in a concave area is called the height of the concave area.*

Definition 24 *A concave area with height 1 is called a bump area.*

Definition 25 *A concave area with height greater than 1 is called a hill area.*

Corollary 9 *A hill area must contain gate and onhill nodes, and may contain bump nodes.*

4.3.3 Removing Hill Area (RHA) Algorithm

In this section, we present the *Removing Hill Area (RHA)* algorithm, which removes some or all onhill nodes in hill areas. This is to prevent packets entering and retreating the hill areas where the destination node does not reside.

The RHA algorithm

Before introducing the RHA algorithm, it is necessary to define the *initiator of a concave area*.

Definition 26 *The concave node, whose identification equals to the identification of the concave area that it resides in, is called the initiator of the concave area.*

Corollary 10 *Each concave node in a concave area knows the identification of the initiator of the concave area, which is the same as the value of variable CId.*

The proposed *RHA* algorithm tries to remove some or all onhill nodes of hill areas. However, the virtual removal of onhill nodes must keep the connectivity of the remaining network, which is to guarantee the correctness of the proposed routing algorithm that will be explained later.

Let N denote the original connected network. The detail of the proposed *RHA* algorithm is described in Algorithm 6.

1. All onhill nodes are marked as *removed* initially.
2. A gate node, say g, sends a *Gate* message, which contains its identification, Id_g, to the initiator of the hill area, say *HA*, that it resides in, say node i, whose identification equals to CId_g.
3. Upon receiving the *Gate* message from g, node i records Id_g contained in the message.
4. From the recorded identification of all gate nodes of *HA* that sent the *Gate* message to it, node i selects one as the *anchor gate node* of the hill area *HA* (i.e., the one with maximum identification), say node a.
5. Node i sends an *AllGates* message to node a, which contains the identifications of all the gate nodes in *HA* that it recorded.
6. Upon receiving the *AllGates* message, node a sends a *KeepConnectivity* message to each gate node of *HA*.
7. Upon receiving a *KeepConnectivity* message, a node, say u, marks itself as *added*. It records the previous hop node of the message, say p, selects a next hop node, say n, and keeps the edge incident on nodes u and p that is denoted by up, and the edge incident on nodes u and n that is denoted by un in a *Added Edge (AE)* set of it.

Algorithm 6: *The RHA algorithm*

Note that each message in Algorithm 6 is routed using *GPSR(N)*, which means that *GPSR* [30] works on network N.

In the proposed RHA algorithm, each onhill node is marked as *removed* initially. From Corollary 10, each gate node of HA knows the identification of the initiator of HA, say node i, which equals to the value of the variable CId.

When a gate node sends a *Gate* message to i, node i knows the identification of all gate nodes of HA, from which i selects one as an anchor gate node of HA, say node a. Node i sends an *AllGate* message to node a, which causes node a to know the identification of all the gate nodes in HA. Node a sends the *KeepConnectivity* messages to all other gate nodes in HA, which helps node a to find routes to all of them.

The nodes in the routes that include onhill nodes are marked as *added*. The edges in the routes are kept in the *Added Edge (AE)* set of each node marked as *added*. The onhill nodes that are still marked as *removed* are removed by the algorithm RHA.

The RHA graph

In this section, we give the definition of the *RHA* graph and prove that it is a connected graph.

Definition 27 *All ground and foothill nodes, and onhill nodes marked as added in the RHA algorithm, are called RHA nodes.*

Definition 28 *All edges between ground and foothill nodes, and edges in the Added Edge (AE) set of each node marked as added in the RHA algorithm are called RHA edges.*

Definition 29 *The graph that contains all RHA nodes and RHA edges is called the RHA graph, denoted by G_{RHA}.*

Definition 30 *We define the union of two graphs, say graphs X snd Y, as a graph that consists of all nodes and edges contained in X and all nodes and edges contained in Y, and is denoted by $X \cup Y$.*

Lemma 15 *G_{RHA} generated by the RHA algorithm is a connected graph when there is only one hill area in the original network.*

Proof: Let N denote the graph of the original network that is connected and HA denote the hill area in the original network.

We divide graph N to two subgraphs as shown in Figure 4.5, one that contains all ground and foothill nodes and all edges between them, which is denoted by N_1, and one that contains all onhill nodes and all edges between them, which is denoted by N_2.

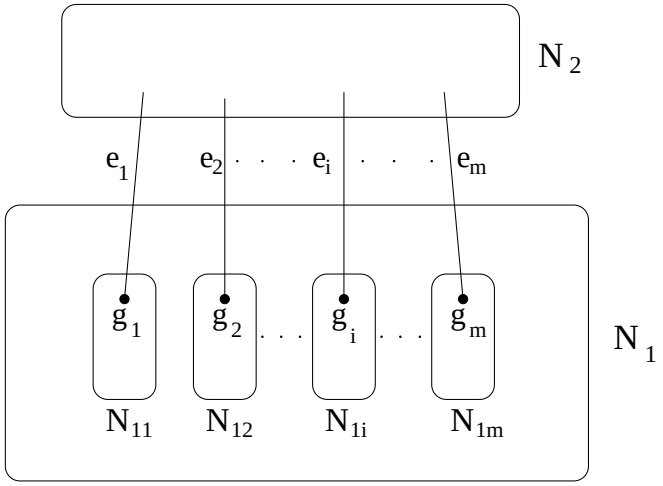


Figure 4.5: A network before the RHA algorithm is applied

Based on Corollary 9, HA must contain gate nodes. Suppose there are $m(m \geq 1)$ gate nodes in HA denoted by $\{g_i \mid 1 \leq i \leq m\}$. Based on Corollary 7 and Corollary 6, N_1 and N_2 are connected only by the edges between g_i and the corresponding onhill nodes with height 2 in N_2, which are denoted by e_i, where $1 \leq i \leq m$ as shown in Figure 4.5. This makes N equal to $N_1 \cup N_2 \cup \{e_i | 1 \leq i \leq m\}$ based on Definition 30.

In N_1, let $N_{1i}(1 \leq i \leq m)$ denote the graph, the node set of which contains gate node g_i and all ground, bump or gate nodes that can be reached by g_i directly or through nodes that are not onhill nodes, and the edge set of which contains all edges between nodes in the node set of it. We have that $N_{1i}(1 \leq i \leq m)$ is connected.

Let U_{N1i} $(1 \leq i \leq m)$ denote $N_{11} \cup N_{12} \cup \cdots \cup N_{1m}$. Because U_{N1i} $(1 \leq i \leq m)$ contains all gate nodes, it contains all ground and foothill (i.e., bump or gate) nodes, based on Corollary 8.

Suppose between node u_i in network N_{1i} and node u_j in network N_{1j}, there is an edge u_iu_j, where $1 \leq i \leq m, 1 \leq j \leq m$ and $i \neq j$. Then edge u_iu_j must be contained in both N_{1i} and N_{1j}. Therefore, U_{N1i} $(1 \leq i \leq m)$ contains all edges between ground and foothill nodes. As a result, $N_1 = U_{N1i}$ $(1 \leq i \leq m)$.

In the RHA algorithm, all onhill nodes are removed initially, which causes graph N_2 and edges $\{e_i | 1 \leq i \leq m\}$ to be removed as shown in Figure 4.6. This may cause the remaining graph N_1 to be partitioned if any two graphs among $N_{11}, N_{12}, \cdots, N_{1m}$, are not connected.

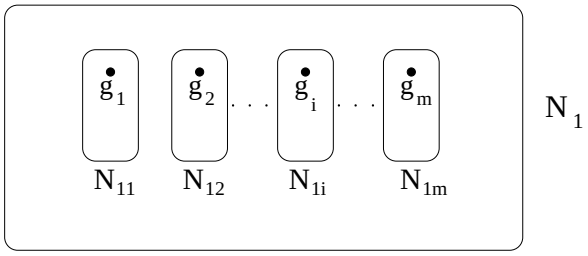


Figure 4.6: A network at the beginning of the RHA algorithm

The RHA algorithm selects a gate node, say g_1, as an anchor gate node, and makes g_1 find routes to all other gate nodes, i.e., routes $R_2, R_3, \cdots, R_m$ to $g_2, g_3, \cdots, g_m (m \geq 2)$, respectively, as shown in Figure 4.7, where R_i $(2 \leq i \leq m)$ contains all nodes and all edges in the route from g_1 to g_i. Let $R = R_2 \cup R_3 \cup \cdots \cup R_m$. This makes the graph of $N_1 \cup R$ connected.

Based on the RHA algorithm as shown in Algorithm 6, all nodes in R are marked as *added* and all edges in R are kept in the *Added Edge (AE)* set of each node in R. Because N_1 contains all ground and foothill nodes and all edges between them, we have G_{RHA} equals to the graph of $N_1 \cup R$, based on Definition 29. Thus, the lemma follows. □

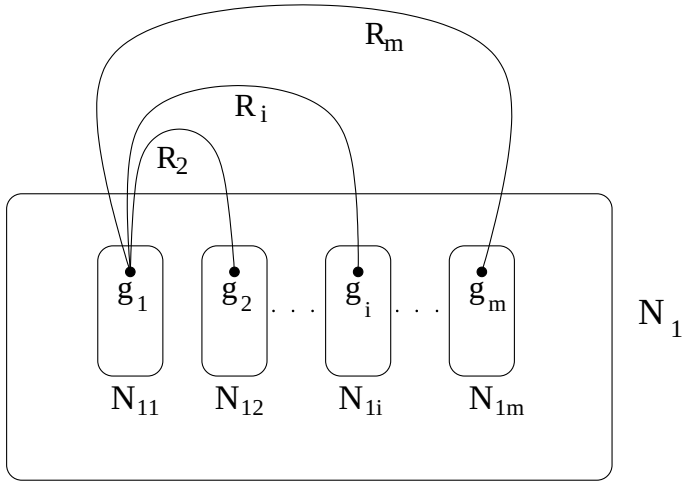


Figure 4.7: A network after the RHA algorithm is applied

Lemma 16 *Graph G_{RHA} generated by the RHA algorithm is a connected graph when there is more than one hill area in the original network.*

Proof: Let N denote the graph of the original network that is connected. Suppose there are $n(n \geq 2)$ hill areas in the original network, which are denoted by $HA_1, HA_2, \cdots, HA_n$.

We divide graph N into two subgraphs as shown in Figure 4.8. First, the graph that contains all ground and foothill nodes and all edges between them, which is denoted by N_1. The second, the graph that contains all onhill nodes and all edges between them, which is denoted by N_2.

Based on Corollary 9, a hill area HA_i $(1 \leq i \leq n)$ must contain gate nodes. For simplicity, suppose that there are $m(m \geq 1)$ gate nodes in each hill area. Let g_{ij} $(1 \leq j \leq m)$ denote a gate node in hill area HA_i. Based on Corollary 7 and Corollary 6, N_1 and N_2 are connected only by the edges between g_{ij} and the corresponding onhill nodes with height 2 in N_2, which are denoted by e_{ij}, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Thus, N equals $N_1 \cup N_2 \cup \{e_{ij} | 1 \leq i \leq n \ \& \ 1 \leq j \leq m\}$.

In N_2, let N_{2i} denote the graph that contains all the onhill nodes in hill area HA_i, where $1 \leq i \leq n$, and all the edges between them. Based on Observation 2, there is no edge between any two nodes in two different graphs among $N_{21}, N_{22}, \cdots, N_{2n}$. Let $U_{N_{2i}}$

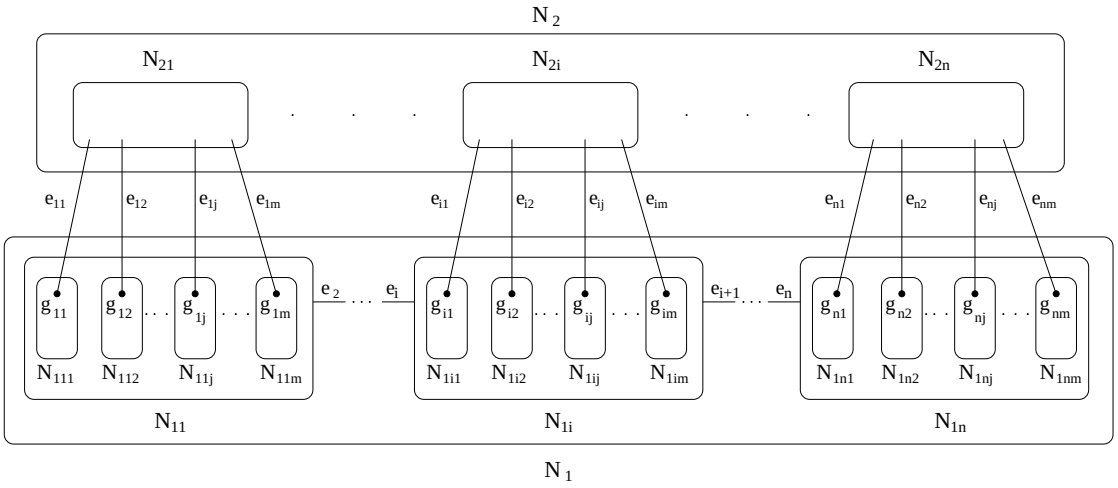


Figure 4.8: A network before the RHA algorithm is applied

$(1 \leq i \leq n)$ denote the graph of $N_{21} \cup N_{22} \cup \cdots \cup N_{2n}$. We have that $N_2 = U_{N_{2i}}$ $(1 \leq i \leq n)$.

In N_1, let N_{1i} denote the graph, the node set of which contains all gate nodes in HA_i $(1 \leq i \leq n)$, which are $\{g_{ij} | 1 \leq j \leq m\}$, and all ground or foothill nodes that can be reached by $\{g_{ij} | 1 \leq j \leq m\}$ directly, or through nodes that are not onhill nodes, and the edge set of which contains all edges incident on nodes in the node set of it. Let $U_{N_{1i}}$ $(1 \leq i \leq n)$ denote the graph of $N_{11} \cup N_{12} \cup \cdots \cup N_{1n}$, we have that $U_{N_{1i}}$ $(1 \leq i \leq n)$ contains all ground and foothill (i.e., bump and gate) nodes in the network, based on Corollary 8.

Suppose between node u_{1i} in network N_{1i} and node u_{1j} in network N_{1j}, there is an edge $u_{1i}u_{1j}$, where $1 \leq i \leq n, 1 \leq j \leq n$, and $i \neq j$. Then edge $u_{1i}u_{1j}$ must be contained in both N_{1i} and N_{1j}. Therefore, $N_1 = U_{N_{1i}}$ $(1 \leq i \leq n)$.

From above, we have $N = U_{N_{1i}} \cup U_{N_{2i}} \cup \{e_{pj} | 1 \leq p \leq n, 1 \leq j \leq m\}$, where $1 \leq i \leq n$. Because N is connected, there must be edges between graphs among $N_{11}, N_{12}, \cdots, N_{1n}$ to make them connected to each other. Suppose there is an edge between $N_{1(i-1)}$ and N_{1i}, as shown in Figure 4.8, which is denoted by e_i, where $2 \leq i \leq n$. This makes $N_1 = U_{N_{1i}} \cup \{e_k | 2 \leq k \leq n\}$ $(1 \leq i \leq n)$.

In graph $N_{1i}(1 \leq i \leq n)$, let $N_{1ij}(1 \leq j \leq m)$ denote the graph, the node set of which contains the gate node g_{ij} in a hill area HA_i and all the ground, bump or gate nodes that can be reached by g_{ij} directly or through nodes that are not onhill nodes, and the edge set of which contains all edges incident on nodes in the node set of it. We have that $N_{1ij}(1 \leq i \leq n, 1 \leq j \leq m)$ is connected.

Let $U_{N_{1ij}}$ $(1 \leq j \leq m)$ denote the graph of $N_{1i1} \cup N_{1i2} \cup \cdots \cup N_{1im}$. Based on the proof of Lemma 15, we have $N_{1i} = U_{N_{1ij}}$ $(1 \leq j \leq m)$.

In the RHA algorithm, all onhill nodes are removed initially, which causes graph N_2 and edges $\{e_{ij}|1 \leq i \leq n, 1 \leq j \leq m\}$ to be removed. This may cause the remaining graph N_1 to be partitioned if any two graphs among $N_{1i1}, N_{1i2}, \cdots, N_{1im}$ that are contained in N_{1i} $(1 \leq i \leq n)$ are not connected as shown in Figure 4.9.

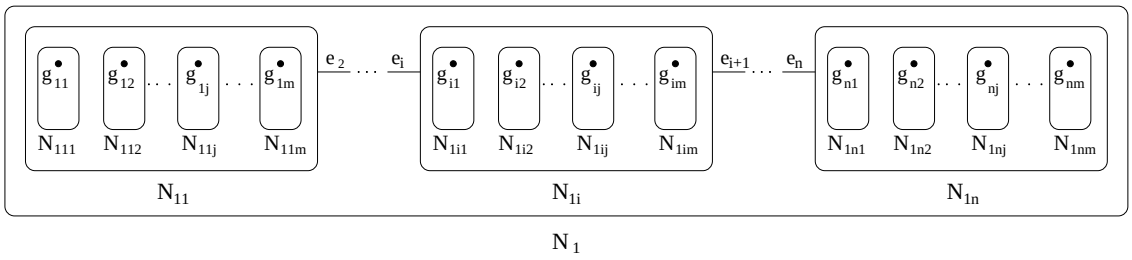


Figure 4.9: A network at the beginning of the RHA algorithm

The RHA algorithm selects a gate node, say g_{i1}, in $N_{1i}(1 \leq i \leq n)$, as an anchor gate node, and makes g_{i1} find routes to all other gate nodes in N_{1i}, i.e., routes $R_{i2}, R_{i3}, \cdots, R_{im}$ to $g_{i2}, g_{i3}, \cdots, g_{im}(m \geq 2)$, respectively, as shown in Figure 4.10, where R_{ij} $(2 \leq j \leq m)$ contains all nodes and all edges in the route from g_{i1} to g_{ij}.

Let $R_i = R_{i2} \cup R_{i3} \cup \cdots \cup R_{im}$ $(1 \leq i \leq n)$. Based on the proof of Lemma 15, we have that graph $N_{1i} \cup R_i$ $(1 \leq i \leq n)$ is connected. Let $R = R_1 \cup R_2 \cup \cdots \cup R_n$. This makes $U_{N_{1i}} \cup R \cup \{e_k|2 \leq k \leq n\} = N_1 \cup R$ connected, where $1 \leq i \leq n$.

Based on the RHA algorithm shown in Algorithm 6, all nodes in R are marked as *added* and all edges in R are kept in the *Added Edge (AE)* set of each node in R. Because N_1 contains all ground and foothill nodes and all edges between them, G_{RHA} is the same

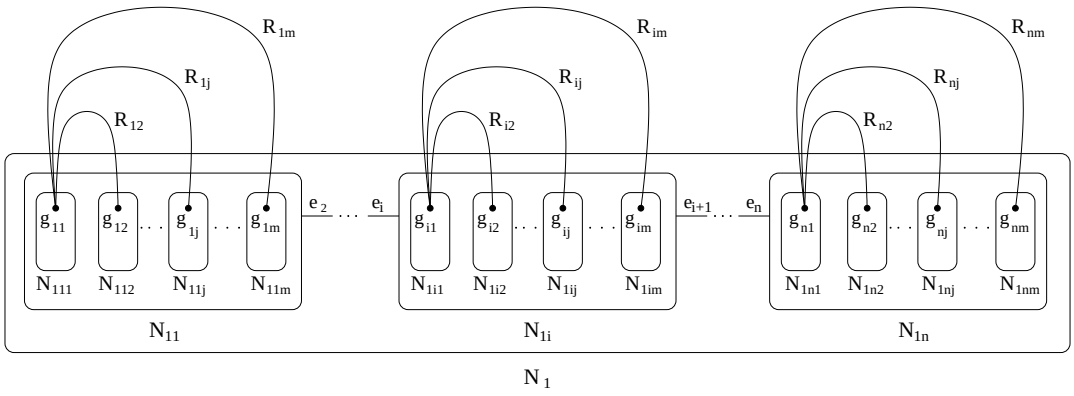


Figure 4.10: A network after the RHA algorithm is applied

as the graph of $N_1 \cup R$. Thus, the lemma follows. □

Theorem 7 *G_{RHA} generated by the RHA algorithm is a connected graph when there are one or more hill areas in the original network.*

Proof: Based on Lemma 15 and Lemma 16, the theorem follows. □

$RHA_{\bar{i}}$ graph

Suppose there are $n(n \geq 1)$ hill areas, denoted by $HA_1, HA_2, \cdots, HA_n$, in the original network. Let $RHA_{\bar{i}}$ graph be a graph generated by removing some or all onhill nodes from all hill areas except HA_i. This is for the later use in routing when the destination node is in the hill area HA_i. Next, we introduce the definition of the $RHA_{\bar{i}}$ graph in detail as follows:

Definition 31 *All ground and foothill nodes, all onhill nodes in a hill area, $HA_i(1 \leq i \leq n)$, and all onhill nodes marked as added in other hill areas except HA_i are called $RHA_{\bar{i}}$ nodes.*

Definition 32 *Edges between all ground and foothill nodes in N, and all onhill nodes in a hill area, $HA_i(1 \leq i \leq n)$, and edges in the AE set of each node marked as added in other hill areas except HA_i are called $RHA_{\bar{i}}$ edges.*

Definition 33 *The network graph, which contains all $RHA_{\bar{i}}$ nodes and all $RHA_{\bar{i}}$ edges, is called the $RHA_{\bar{i}}$ graph, denoted by $G_{RHA_{\bar{i}}}(1 \leq i \leq n)$.*

Theorem 8 *Suppose there are $n(n \geq 1)$ hill areas in the original network. Then graph $G_{RHA_{\bar{i}}}(1 \leq i \leq n)$ is a connected graph.*

Proof: Let N denote the graph of the original network that is connected. Suppose there are $n(n \geq 1)$ hill areas in the original network, which are denoted by $HA_1, HA_2, \cdots, HA_n$.

When n equals to 1, $G_{RHA_{\bar{i}}} = G_{RHA_{\bar{1}}} = N$, which is connected.

When $n > 1$, we use the same denotations as those in the above proof of Lemma 15. Let U_{R_i} $(1 \leq i \leq n)$ denote $R_1 \cup R_2 \cup \cdots \cup R_n$. We have $G_{RHA_{\bar{i}}} = U_{N_{1j}} \cup U_{R_j} \cup \{e_k | 2 \leq k \leq n\} \cup N_{1i} \cup N_{2i} \cup \{e_{ip} | 1 \leq p \leq m\}$ as shown in Figure 4.11, where $1 \leq j \leq n$ and $j \neq i$.

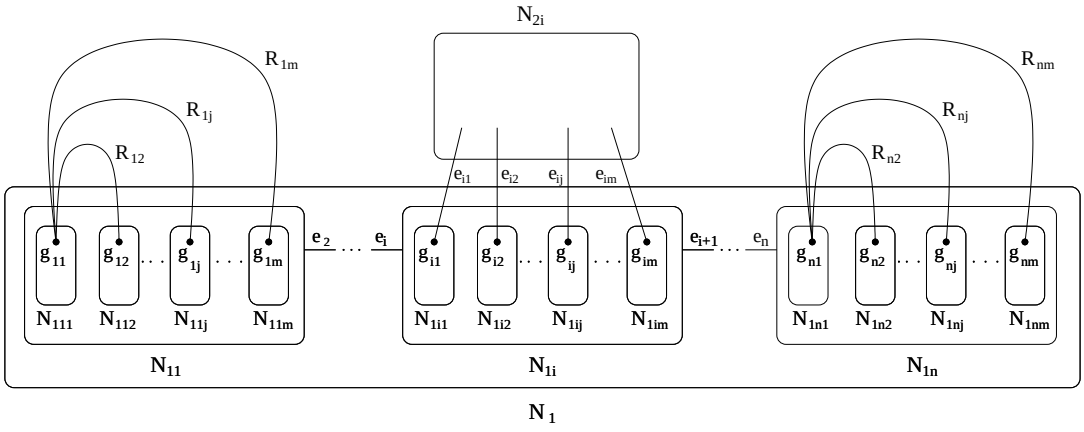


Figure 4.11: A network after the RHA algorithm is applied

Based on Definition 25, Corollary 9 and Observation 1, we have $N_{1i} \cup N_{2i} \cup \{e_{ip} | 1 \leq p \leq m\}$ is connected. Because $N_{1i} \cup R_i (1 \leq i \leq n)$ is connected, the theorem follows. □

4.3.4 Hill-Area-Restricted GPSR (HAR-GPSR) Routing Algorithm

In this section, we present the Hill-Area-Restricted GPSR (HAR-GPSR) routing algorithm, which is based on the existing Greedy-Face-Greedy (GFG) geographic routing protocol,

GPSR [30]. We also prove that the *HAR-GPSR* guarantees the packet delivery, which means that it can always find a route to a destination in a connected network.

Let N denote the original connected network. The source and destination nodes are denoted by nodes s and d, respectively. Suppose each source node knows the virtual coordinate and the Concave area Identification (CId) of the destination node. The packet generated by s that needs to be routed to d is denoted by $P\{(d_x, d_y, d_z), CId_d, RT\}$, where the first field (d_x, d_y, d_z) is the virtual coordinate of node d, the second field CId_d is the identification of the concave area that d may reside in, which is -1 if d does not reside in any concave area, and the third field RT is the current routing type. The details of the algorithm HAR-GPSR are given in Algorithm 7.

The proposed HAR-GPSR algorithm contains two steps. Step 1 is to set the RT field contained in packet P, which denotes the routing type of P. There are three kinds of routing types, which are $GPSR(N)$, $GPSR(G_{RHA})$, and $DownHill$.

When generating a packet P, source node s sets the RT field as follows:

If s is a ground or bump node, or if s is a gate or onhill node and the destination node d is an onhill node in the same concave area as s resides in, it sets the routing type as $GPSR(N)$ in the RT field. Otherwise, if s is a gate node, it sets the routing type as $GPSR(G_{RHA})$ in the RT field. If s is an onhill node, it sets the routing type as $DownHill$ in the RT field.

Let u denote an intermediate forwarding node of packet P. If u is a ground, bump or onhill node, it keeps the RT field contained in P. If u is a gate node, it checks whether the RT field contained in P is $DownHill$. If yes, it means that the destination node is not in the same concave area as u resides in. In this case, u sets the RT field as $GPSR(G_{RHA})$. Otherwise, it means that the RT field in packet P is $GPSR(N)$ or $GPSR(G_{RHA})$. In this case, it checks if the destination node d is an onhill node in the same concave area as it resides in. If yes, it sets the RT field as $GPSR(N)$. Otherwise, it sets the RT field as $GPSR(G_{RHA})$.

1. **Step 1: Setting the RT field in a packet P**

   When generating a packet P, node s sets the RT field of P as follows:

   i. If s is a ground or bump node, it sets RT as *GPSR(N)*.

   ii. If s is a gate node, it checks if $d_z \geq 2$ and $CId_d = CId_s$. If so, it sets RT as *GPSR(N)*. Otherwise, it sets RT as *GPSR(G_{RHA})*.

   iii. If s is an onhill node, it checks if $d_z \geq 2$ and $CId_d = CId_s$. If so, it sets RT as *GPSR(N)*. Otherwise, it sets RT as *DownHill*.

   When receiving a packet P, a node, say u, sets the RT field of P as follows:

   iv. If u is a ground or bump node, it keeps the RT field contained in P.

   v. If u is a gate node, it checks if the RT field contained in P is *DownHill*. If so, it sets RT as *GPSR(G_{RHA})*.

   Otherwise, it means the RT field contained in P is *GPSR(N)* or *GPSR(G_{RHA})*, it further checks if $d_z \geq 2$ and $CId_d = CId_u$. If so, it sets the RT field as *GPSR(N)*. Otherwise, it sets the RT field as *GPSR(G_{RHA})*.

   vi. If u is an onhill node, it keeps the RT field contained in P.

2. **Step 2: Routing a packet P**

   Based on the RT field contained in P, a node, say u, routes P as follows:

   i. If $RT =$ *GPSR(N)*, it routes P using *GPSR* on the original network N.

   ii. If $RT =$ *GPSR(G_{RHA})*, it routes P using *GPSR* on the RHA graph, G_{RHA}.

   iii. If $RT =$ *DownHill*, it forwards packet P to a neighbor node, say v, which satisfies $CId_v = CId_u$ and $v_z < u_z$.

Algorithm 7: *The HAR-GPSR routing algorithm*

Step 2 of the HAR-GPSR algorithm is to route packet P based on the routing type in the RT field set in Step 1. If the routing type is $GPSR(N)$, P will be forwarded using $GPSR$ on the original network N. If the routing type is $GPSR(G_{RHA})$, P will be forwarded using $GPSR$ on the RHA graph, G_{RHA}. If the routing type is $DownHill$, the forwarding node will transfer P to a neighbor node, which is in the same concave area as it resides in and has a smaller z coordinate than its own.

Theorem 9 *HAR-GPSR always finds a route to a destination in a connected network.*

Proof: Let N denote the original network that is connected. Suppose a source node s generates a packet P that needs to be routed to a destination node d.

Suppose there is no hill area in N. Based on the routing algorithm HAR-GPSR shown in Algorithm 7, P will be routed using *GPSR* on network N that is connected. Because *GPSR* finds all existing routes (that is, on network graphs where the destination is connected) [30], the theorem follows.

Suppose there are $n(n \geq 1)$ hill areas in N, which are denoted by HA_i $(1 \leq i \leq n)$. There are two cases based on the location of the destination node d as follows:

Case 1: d is not in any hill area. In this case, if the source node s is a ground, bump or gate node, HAR-GPSR will route P using *GPSR(G_{RHA})*, which means using *GPSR* on *RHA* graph G_{RHA}.

If s is an onhill node, packet P will be routed down the hill area that s resides in to a gate node of the hill area, and HAR-GPSR will route P using *GPSR(G_{RHA})*.

Case 2: d is in a hill area, say HA_1. In this case, if the source node s is a ground, bump or gate node, HAR-GPSR will route P using *GPSR($G_{RHA_{\overline{1}}}$)*, which means using *GPSR* on $RHA_{\overline{1}}$ graph $G_{RHA_{\overline{1}}}$.

If s is an onhill node that resides in a hill area $HA_i(1 \leq i \leq n)$, it further checks if HA_i is the same as HA_1. If so, the packet P will be routed using *GPSR(N)*, until it reaches the destination node d, which means the routing succeeds.

If this is not the case, packet P will be routed down the hill area HA_i to a gate node in HA_i, and HAR-GPSR will route P using *GPSR($G_{RHA_{\overline{1}}}$)*.

Based on Theorem 7 and Theorem 8, we have that G_{RHA} and $G_{RHA_{\overline{1}}}$ are connected network graphs. Because *GPSR* finds all existing routes (that is, on network graphs where the destination is connected) [30], *GPSR(G_{RHA})* and *GPSR($N_{RHA_{\overline{1}}}$)* can always find a route to d in both of the above two cases. Thus, the theorem follows. □

4.4 Simulation Study and Analysis

4.4.1 Simulation Settings

In our simulations, the network was a square area of $2000 \times 2000m^2$. We simulated two environments termed as *small void* and *dominant void*, respectively, as in NEAR [4]. For the *small void* environment, we set a void in the middle of the square area of the network that covered approximately 12% of the network size. In addition, a peninsula entered the *small void* as shown in Figure 4.12. The coordinate of nodes A, B, C, D, E, F, G and H were (500, 700), (800, 700), (800, 1000), (1200, 1000), (1200, 700), (1500, 700), (1500, 1300) and (500, 1300), respectively.

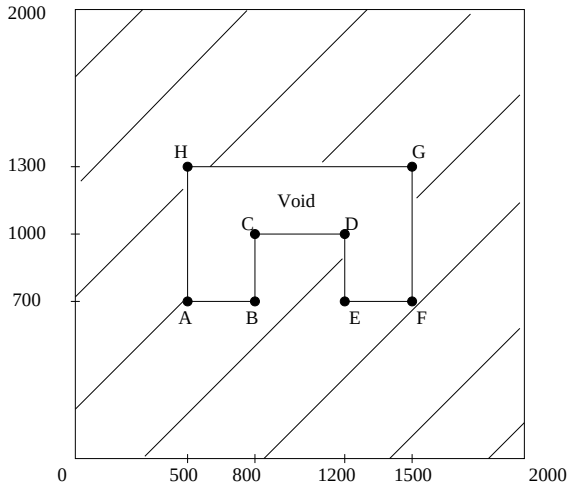


Figure 4.12: Simulation Environment of *Small Void*

For the *dominant void* environment, we set a void in the middle of the square area of

the network that covers approximately 25% of the network size. In addition, a peninsula entered the *dominant void* as shown in Figure 4.13. The coordinate of nodes A, B, C, D, E, F, G and H were (400, 500), (800, 500), (800, 1000), (1200, 1000), (1200, 500), (1600, 500), (1600, 1500) and (400, 1500), respectively.

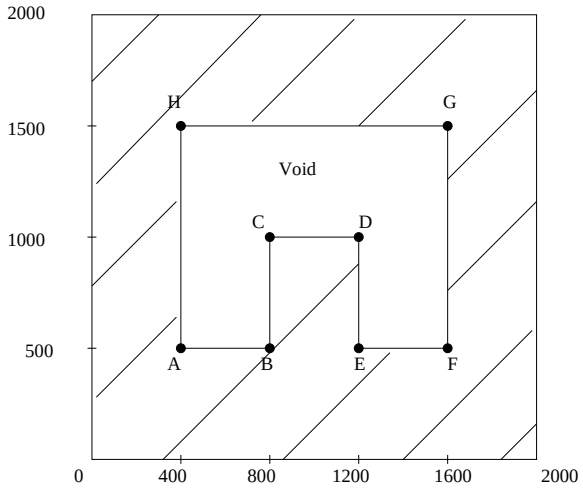


Figure 4.13: Simulation Environment of *Dominant Void*

In both *small void* and *dominant void* environments, nodes were distributed randomly in the network except the void area, as shown in the shaded area in Figure 4.12 and Figure 4.13, respectively. Each node has a transmission range of 250 meters. Nodes in the network construct a *unit-disk* graph, where there is an edge incident on two nodes if they are within each other's transmission range.

The number of nodes in the network was changed from 340 to 600. By changing the number of nodes in the network, we changed the network density or the average node degree (the number of neighbors of a node). In the simulation study, we generated seven network topologies randomly, which contained $340, 380, 420, 450, 500, 550$ and 600 nodes, respectively. The approximate average node degrees were $16, 18, 20, 22, 24, 26$ and 28, respectively. For each topology, we ran simulations 30 times with different seeds. Each value in the following graphs is the average value of the 30 runs.

4.4.2 Simulation Model

In the simulation study, we run the three algorithms of the proposed *HAR* protocol on the current snapshot of the network. At the network start-up, nodes send the information of their physical and virtual coordinates, and the identification of the concave areas where they may reside, in *Hello* messages in a randomly selected order.

When receiving *Hello* messages from its neighbors, a node runs the *Concave Area Identification (CAI)* algorithm, which may update its virtual coordination or the identification of the concave area that it may reside in. In this case, it sends a *Hello* message to notify its neighbors about the update. When no more nodes in the network send the update in *Hello* messages, the *CAI* algorithm converges.

Then, we run the *Removing Hill Area (RHA)* algorithm, at the beginning of which each gate node sends its real and virtual coordinates and the identification of the concave area where it resides in a *Gate* message. It is forwarded to the *initiator* of the concave area that it resides in. When no more nodes in the network send any messages, the *RHA* algorithm converges.

With the information obtained by message exchanging in the *CAI* and *RHA* algorithms, we are ready to run the *HAR-GPSR* routing algorithm. We randomly selected 30% nodes in a network topology as source nodes and for each of them we randomly selected a destination node. At the beginning of the *HAR-GPSR* algorithm, a source node initiates a route discovery by sending a routing message. When no more nodes in the network send any messages, the *HAR-GPSR* algorithm converges.

For each algorithm, we collected the data, i.e., total number of messages broadcast by all nodes, total size of messages in bytes broadcast by all nodes, and etc., during the time interval from the first node starting to send a message to the time when no more nodes send any messages in the network.

4.4.3 Communication Cost of the First Algorithm of Protocols NEAR and HAR

The proposed HAR routing protocol contains three algorithms, which are the CAI algorithm, the RHA algorithm and the HAR-GPSR routing algorithm. The NEAR protocol also contains three algorithms, which are the node reposition algorithm, the void bypass algorithm and the routing algorithm. In both the NEAR and the HAR routing protocols, results of the first two algorithms are the inputs of the third one.

In this section, we use two metrics, the number of messages broadcast by all nodes in the network and the average message size to evaluate the communication cost of the first algorithm of protocols NEAR and HAR. The threshold angle α in the CAI algorithm in HAR is set as $11\pi/9$, which is the same as that in NEAR.

Total number of messages broadcast by all nodes in the network

For the *small void* environment, we plotted the total number of messages broadcast by all nodes with respect to the number of nodes in Figure 4.14. It shows that the total number of messages broadcast by all nodes is 11.3% less in the first algorithm of HAR than that in the first algorithm of NEAR.

For the *dominant void* environment, we plotted the total number of messages broadcast by all nodes with respect to the number of nodes in Figure 4.15. It shows that the total number of messages broadcast by all nodes is 30.3% less in the first algorithm of HAR than that in the first algorithm of NEAR.

Average size of messages broadcast in the network

In the environments of *small void* and *dominant void*, the average size of messages broadcast by all nodes was 68 bytes and 52 bytes in the first algorithm of HAR and NEAR, respectively.

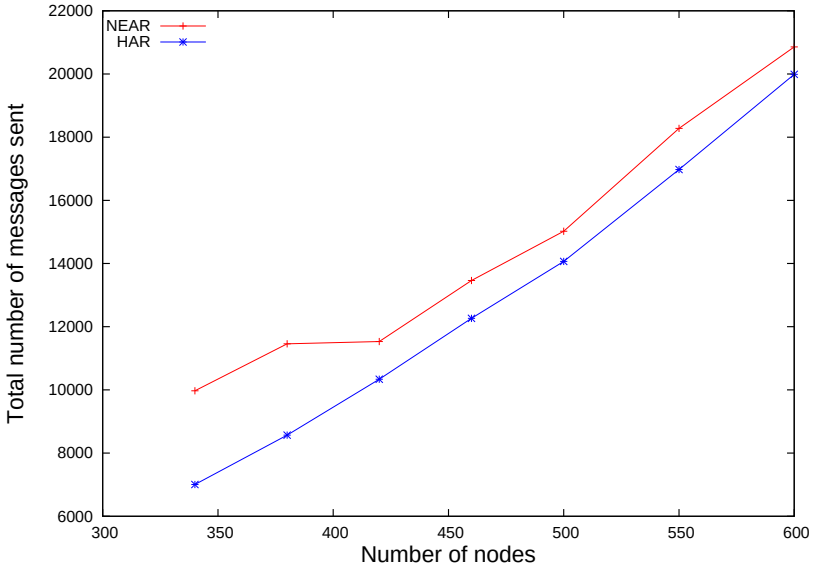


Figure 4.14: Total number of messages broadcast by all nodes in the environment of *small void*

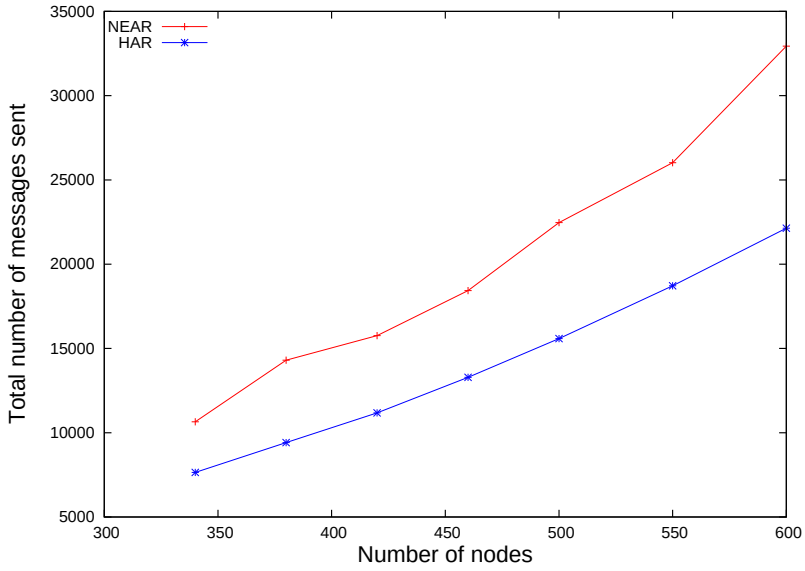


Figure 4.15: Total number of messages broadcast by all nodes in the environment of *dominant void*

Communication cost

Because the average size of messages sent in HAR is slightly larger than that in NEAR, we plotted the communication cost in bytes of the first algorithm of each protocol.

For the environment of *small void*, the communication cost in bytes with respect to the number of nodes are shown in Figure 4.16. It shows that the communication cost of the first algorithm of NEAR is 13.8% less than that of HAR. This is because the average size of messages sent in HAR is slightly larger than that in NEAR.

For the environment of *dominant void*, the communication cost in bytes with respect to the number of nodes are shown in Figure 4.17. It shows that the communication cost of the first algorithm of HAR is 8.9% less than that in NEAR.

4.4.4 Communication cost of the second algorithm of protocols NEAR and HAR

In this section, we use two metrics, the number of messages broadcast by all nodes in the network and the average message size to evaluate the communication cost of the second algorithm of protocols NEAR and HAR. The threshold angle β in NEAR was set as π [4].

Total number of messages broadcast by all nodes in the network

For the environment of *small void*, we depict the total number of messages broadcast by all nodes with respect to the number of nodes in Figure 4.18. We observe that the total number of messages sent in the second algorithm of HAR is 99.9% less than that in the second algorithm of NEAR.

For the environment of *dominant void*, we depict the total number of messages broadcast by all nodes with respect to the number of nodes in Figure 4.19. We observe that the total number of messages sent in the second algorithm of HAR is 99.4% less than that in the second algorithm of NEAR.

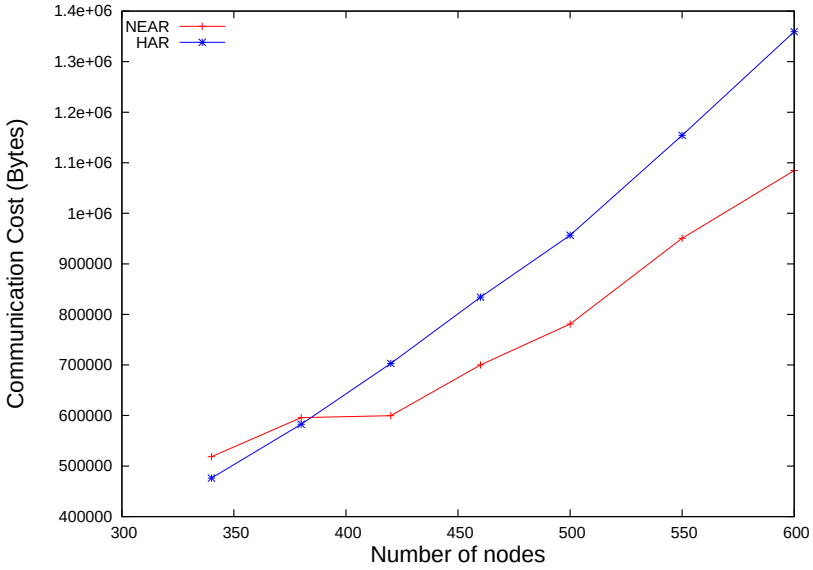


Figure 4.16: Communication cost in the environment of *small void*

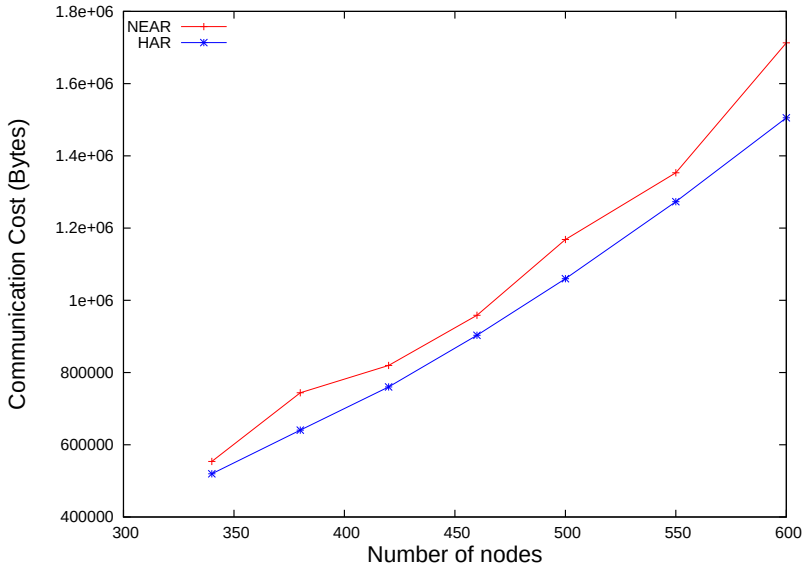


Figure 4.17: Communication cost in the environment of *dominant void*

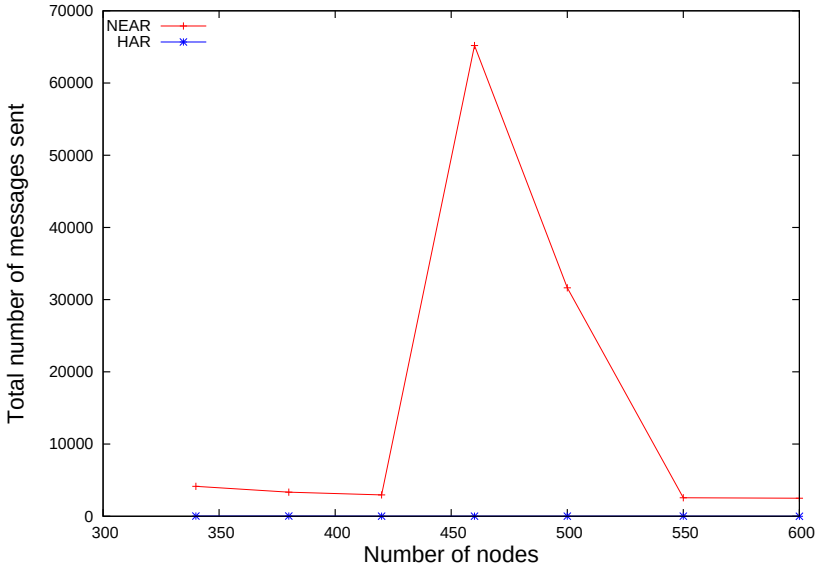


Figure 4.18: Total number of messages broadcast by all nodes in the environment of *small void*

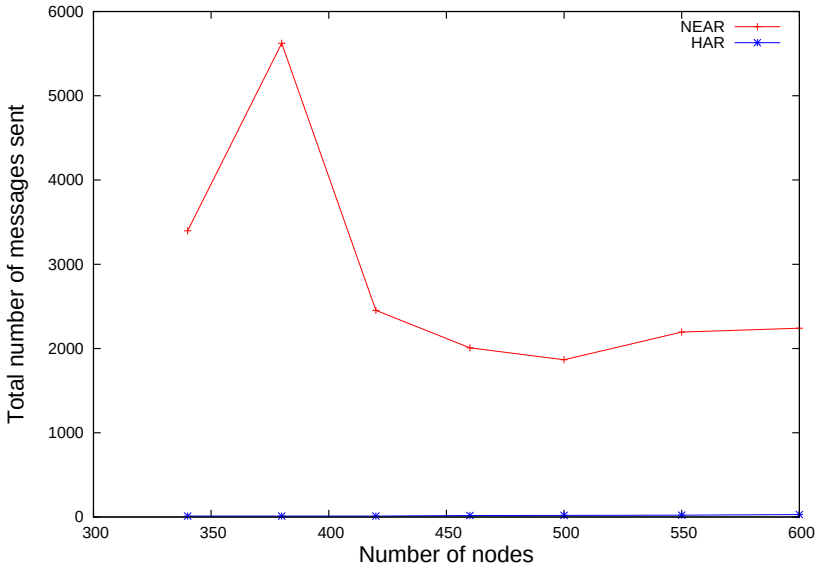


Figure 4.19: Total number of messages broadcast by all nodes in the environment of *dominant void*

Average size of messages broadcast in the network

The average size of messages broadcast by all nodes in the environments of *small void* and *dominant void*, is 68 bytes and 52 bytes in the second algorithm of HAR and NEAR, respectively.

Communication cost

Because the average size of messages sent in HAR is slightly larger than that in NEAR, we plot the communication cost in bytes of the second algorithm of each protocol in this section.

For the environment of *small void*, the communication cost in bytes with respect to the number of nodes are shown in Figure 4.20. It shows that the communication cost in the second algorithm of HAR is 99.9% less than that in the second algorithm of NEAR.

For the environment of *dominant void*, the communication cost in bytes with respect to the number of nodes are shown in Figure 4.21. It shows that the communication cost in the second algorithm of HAR is 99.2% less than that in the second algorithm of NEAR.

4.4.5 Evaluation of the Convergence Time of the First Two Algorithms of NEAR and HAR

In this section, we used the convergence time of the first two algorithms of protocols *NEAR* and *HAR* as another metric to compare their performance.

We assumed the one-hop message transmission latency is T. Namely, it takes an average time T for a message to travel from a node to its neighbors.

We define the convergence time of the first two algorithms of protocols *NEAR* and *HAR* as the time interval, in terms of T, from the first node starting to send messages in the first algorithm to the time that no nodes need to send messages in the second algorithm of each protocol.

For the environment of *small void*, we depicted the convergence time in T with respect to the number of nodes in the network in Figure 4.22. We observe that the convergence

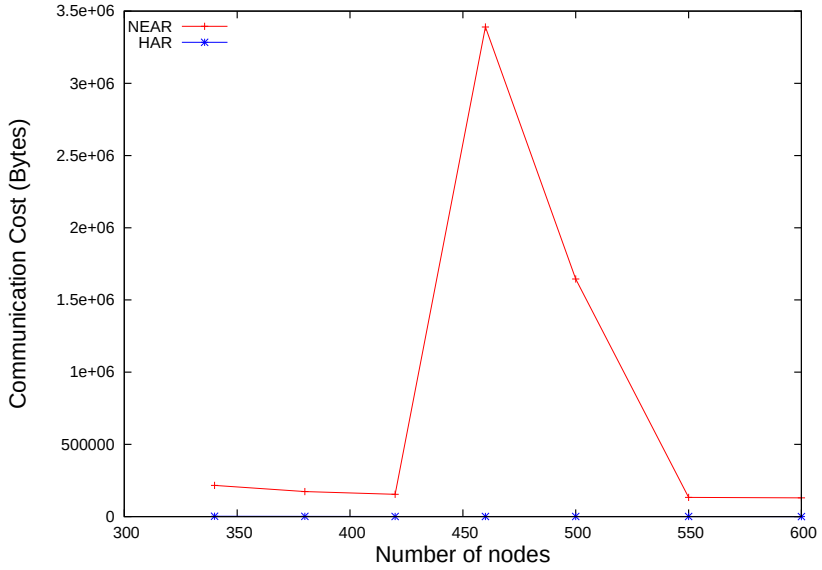


Figure 4.20: Communication cost of NEAR and HAR in the environment of *small void*

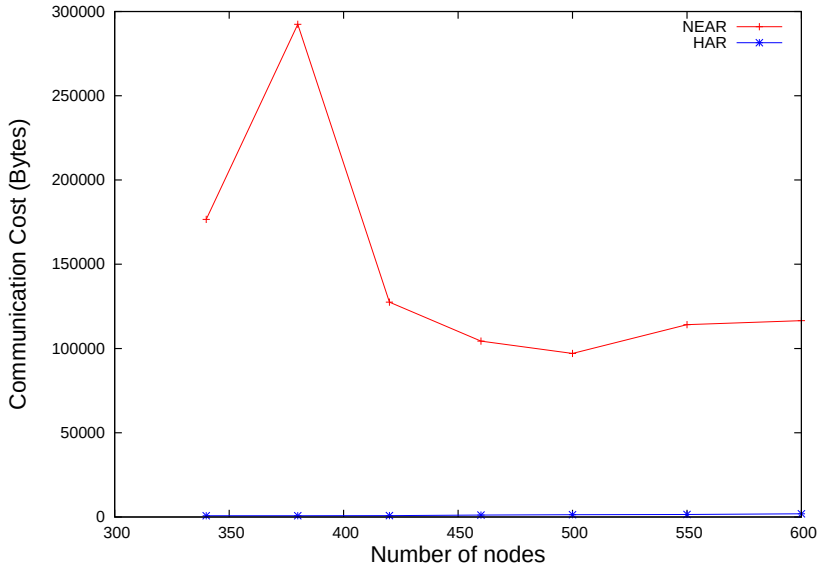


Figure 4.21: Communication cost of NEAR and HAR in the environment of *dominant void*

time of the first two algorithms of *HAR* is 98.6% less than that of the first two algorithms of *NEAR*.

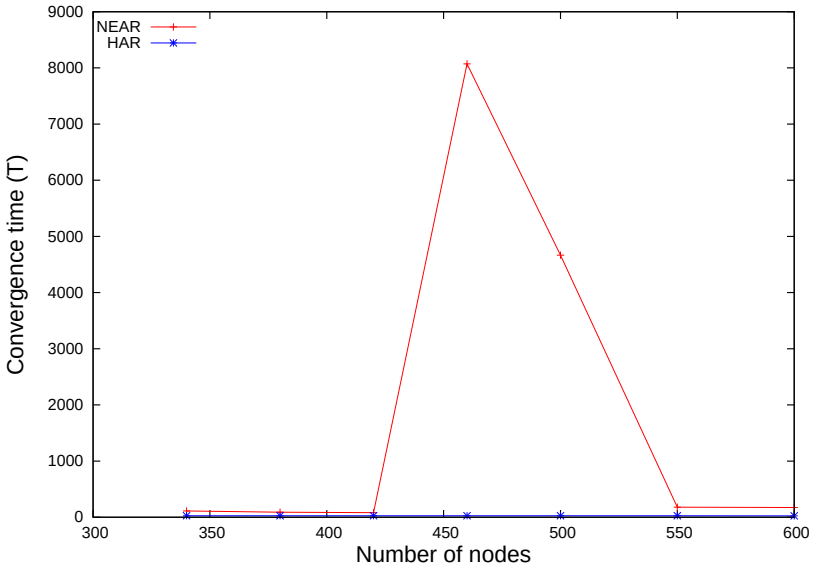


Figure 4.22: Convergence time of NEAR and HAR in the environment of *small void*

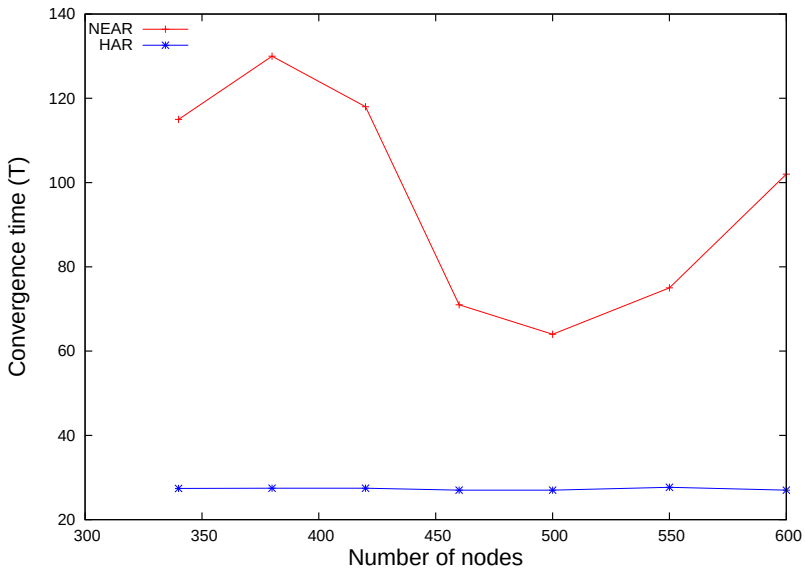


Figure 4.23: Convergence time of NEAR and HAR in the environment of *dominant void*

For the environment of *dominant void*, we depicted the convergence time in T with respect to the number of nodes in the network in Figure 4.23. We observe that the convergence time the first two algorithms of *HAR* is 71.7% less than that of the first two algorithms

of *NEAR*.

4.4.6 Routing Performance

For each network topology, we randomly selected 30% nodes as source nodes and for each we randomly selected a destination node. We depict the routing performance of NEAR, HAR and GPSR in two metrics, the success rate and the average hop count.

Success rate

For the environment of *small void*, we depict the *success rate* with respect to the number of nodes in Figure 4.24. It shows that the success rate of NEAR is 92.7%, and is 100% of both HAR and GPSR. This confirms that the proposed HAR guarantees the packet delivery, while NEAR does not.

For the environment of *dominant void*, we depict the *success rate* with respect to the number of nodes in Figure 4.25. It shows that the success rate of NEAR is 90.6%, and is 100% of both HAR and GPSR. This confirms that the proposed HAR guarantees the packet delivery, while NEAR does not.

Average hop count

Figure 4.24 and Figure 4.25 show that the success rate of routing by both HAR and GPSR are 100% for the environments of both *small void* and *dominant void*. In this section, we first plot the average hop count of routes between all pairs of source and destination nodes by HAR and GPSR with respect to the number of nodes for the environment of *small void* in Figure 4.26. It shows that the average hop count of HAR is 6.7, which is 4.0% less than that of GPSR that is 7.0.

We also plotted the average hop count of routes between all pairs of source and destination nodes by HAR and GPSR with respect to the number of nodes for the environment of *dominant void* in Figure 4.27. It shows that the average hop count of HAR is 8, which is 10.8% less than that of GPSR, which is 9.

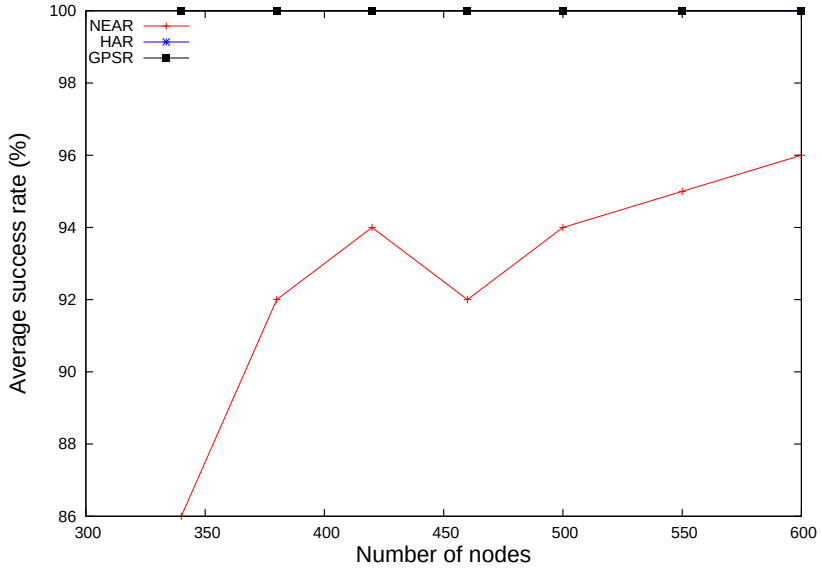


Figure 4.24: Average success rate of NEAR, HAR and GPSR in the environment of *small void*

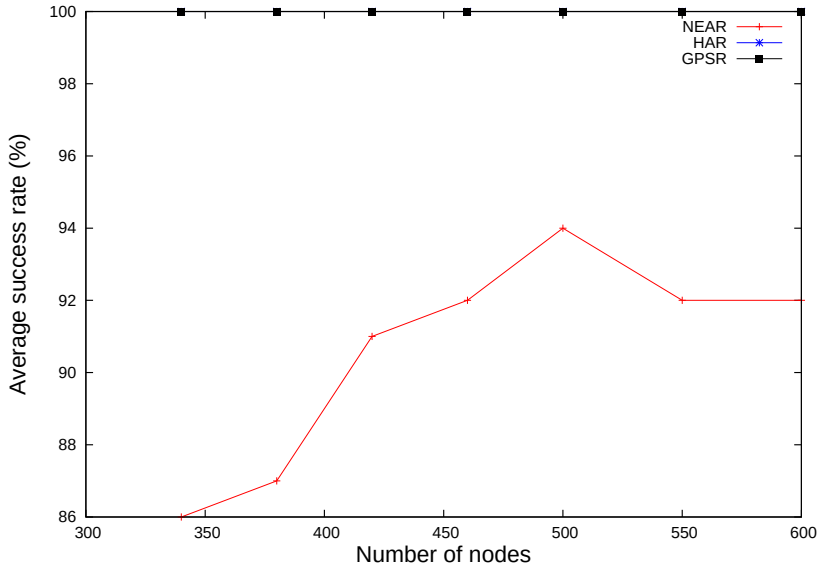


Figure 4.25: Average success rate of NEAR, HAR and GPSR in the environment of *dominant void*

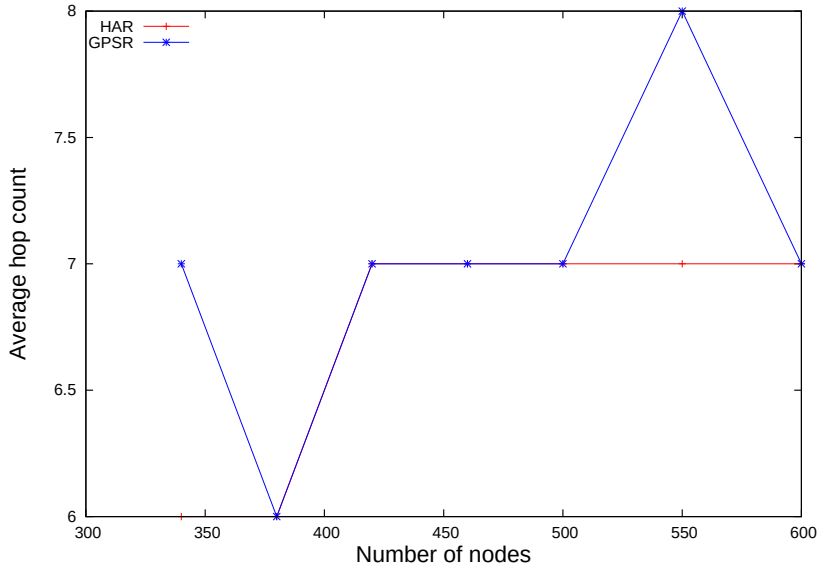


Figure 4.26: Average hop count of HAR and GPSR in the environment of *small void*

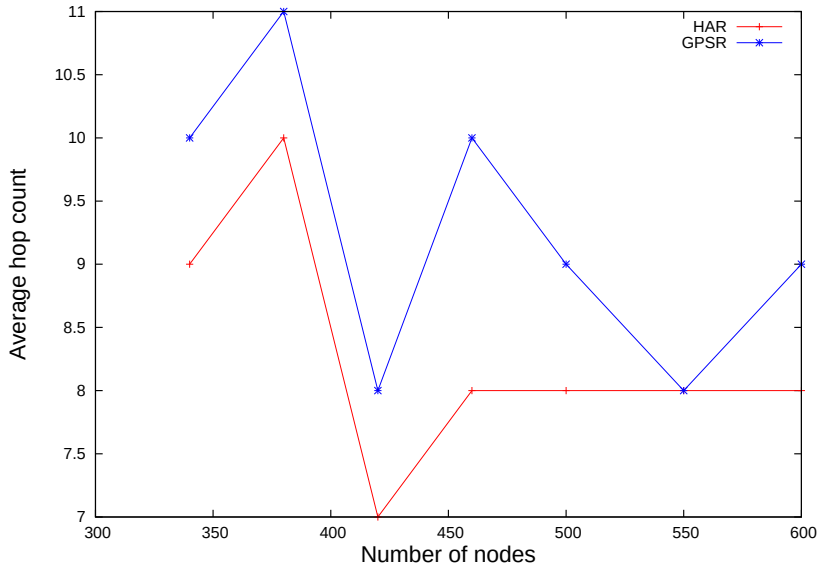


Figure 4.27: Average hop count of HAR and GPSR in the environment of *dominant void*

Figure 4.24 and Figure 4.25 show that the success rates of NEAR in the environments of *small void* and *dominant void* are respectively 92.7% and 90.6%, which means that routes between some pairs of source and destination nodes can not be found successfully by NEAR. Each pair of source and destination nodes, the route between which can be found successfully by NEAR, is called the *NEAR source and destination pair*.

To compare the routing performance in hop count with NEAR for the environment of *small void*, we plotted the average hop count of routes between all *NEAR source and destination pair*s by protocols NEAR, HAR and GPSR with respect to the number of nodes in Figure 4.28. It shows that the average hop count of NEAR is 5.0, 22.2% less than that of GPSR that is 6.4. The average hop count of HAR is 6.3, 2.0% less than that of GPSR.

For the environment of *dominant void*, we plotted the average hop count of routes between all *NEAR source and destination pair*s by protocols NEAR, HAR and GPSR with respect to the number of nodes in Figure 4.29. It shows that the average hop count of NEAR is 6.3, which is 22.8% less than that of GPSR that is 8.1. The average hop count of HAR is 7.6, which is 7.0% less than that of GPSR.

4.5 Related Work

Karp et al. [30] proposed the Greedy perimeter stateless routing (GPSR) for wireless networks. Greedy routing is applied first and perimeter routing is used when greedy routing fails. Greedy routing is applied again when possible. GPSR guarantees the packet delivery. However, extra hops of routing may be needed in networks containing voids.

Zou et al. [70] presented a partial-partition avoiding geographic routing (PAGER) protocol to solve the dead end problem of GFG routing in sensor networks. A dead end node (concave node) is a node that is closer to the base station (BS) than any of its neighbors. PAGER contains two phases, the *shadow spread* phase and *cost spread* phase. It gives each node a forwarding direction based on the cost to the BS of itself and all its neighbors, which as a result avoids packets from arriving at dead end nodes. PAGER works for sensor

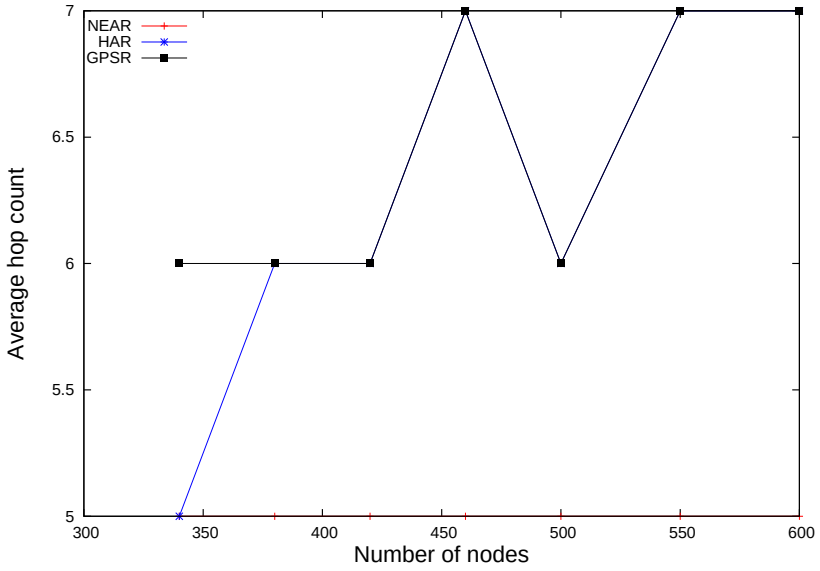


Figure 4.28: Average hop count of NEAR, HAR and GPSR in the environment of *small void*

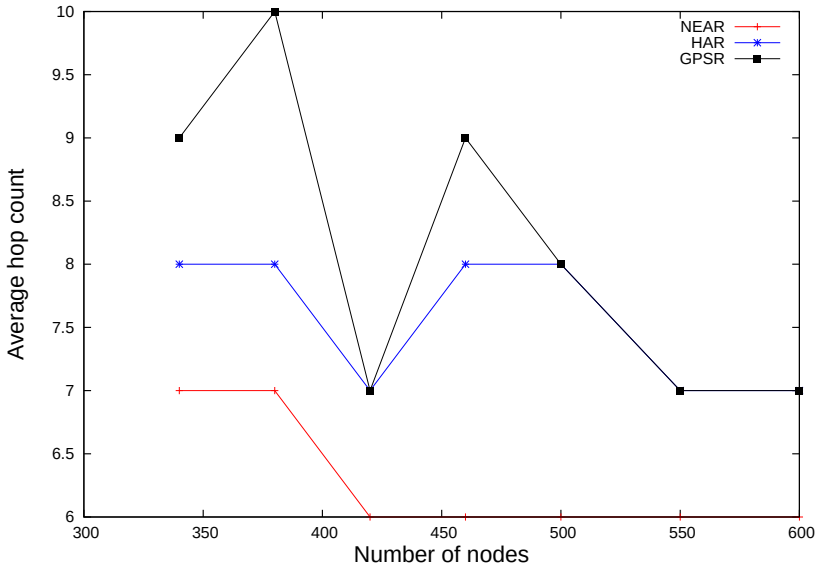


Figure 4.29: Average hop count of NEAR, HAR and GPSR in the environment of *dominant void*

networks, where each node knows the location of the BS.

Arad et al. [4] introduced a node elevation ad hoc routing (NEAR) protocol for mobile ad hoc networks. It contains three algorithms, which are the node repositioning algorithm, void bypass algorithm and routing algorithm. The routing algorithm is based on the GFG routing protocol and the results of the first two algorithms. NEAR improves the routing efficiency in hop count of GFG routing in networks containing voids. However, it does not guarantee the packet delivery and the communication cost of the first two algorithms is high, which is not desirable in mobile ad hoc and sensor networks.

Liu et al. [44] proposed a destination-region-based local minimum aware geometric routing algorithm. Compared to the previous work of *NEAR*, it improves the accuracy of the local minima prediction, which improves routing performance in terms of route length. However, before the routing process, the source node needs to obtain position information, destination region and local minimum area ID of the destination node from the location service. In addition, the local minimum area ID is selected by nodes in the area, which needs cooperation and messages exchange among the nodes.

4.6 Chapter Summary

In Greedy-Face-Greedy routing protocols, greedy routing may take packets to concave nodes, where the perimeter routing is applied to recover from the greedy routing failure. This may cause extra hops of routing by GFG routing protocols in networks containing voids. In this chapter, we proposed a *Hill-Area-Restricted (HAR)* routing protocol, which avoids the extra hops of routing caused by most of the existing GFG routing protocols, and makes it more efficient in hop count. We proved that the proposed HAR guarantees the packet delivery. Compared to the previous work of NEAR, the proposed HAR lowers the communication cost tremendously. Simulation results proved the superior performance of HAR, which is therefore more desirable for mobile ad hoc and sensor networks.

Chapter 5

Concluding Remarks and Directions for Future Research

5.1 Concluding Remarks

Recently, mobile ad hoc and sensor networks have attracted a lot of attention with the widespread emergence of wireless devices. Mobile ad hoc and sensor networks are infrastructureless networks, which consist of wireless and mobile nodes connected in an arbitrary manner without any infrastructure. Because there are no fixed routers, each node acts as both an end system and a router in mobile ad hoc and sensor networks. A node can communicate directly only with nodes in its transmission range. When two nodes are not within each other's transmission range, communication between them requires multi-hop routing and the help of other mobile nodes to route packets between them. Since all nodes are mobile and there is no fixed infrastructure, the design of routing protocols becomes one of the most challenging issues in mobile ad hoc and sensor networks. In addition, nodes usually have limited resources, i.e., memory and power, which motivates the development of efficient routing protocols with low overhead and low bandwidth consumption.

Compared to topology-based routing protocols, position-based (geographic) routing protocols do not need mobile nodes to maintain routing information that is achieved by message flooding. Instead, a node only needs to maintain the location of its neighbors, which is sufficient for it to select the next hop node to route a packet. The low overhead

and bandwidth assumption of geographic routing makes it scalable for mobile ad hoc and sensor networks that contain nodes with limited memory and power.

In mobile ad hoc and sensor networks, most geographic routing protocols, e.g., Greedy-Face-Greedy routing protocols, need nodes to construct planar graphs as the underlying graph for face routing. Idealized and realistic planarization algorithms, which generate a planar subgraph of the original network graph under relatively idealized and realistic environments, have been an active topic of research.

In this dissertation, we developed an idealized planarization algorithm $AlgEcldel$, which can be run by each node distributively with 1-hop neighborhood information to construct an *Edge Constrained Localized Delaunay graph, ECLDel*. We proved that the proposed *ECLDel* is a planar t-spanner of the original *unit-disk* graph, which can be used as the underlying graph for face routing in mobile ad hoc and sensor networks. Compared to the previous algorithm to construct the *PLDel*, our algorithm to construct the *ECLDel* is much simpler and converges faster. This is because we significantly decrease the number and size of messages broadcast by each node in the construction, which results in a far lower communication cost and is more desirable for mobile ad hoc and sensor networks. Our simulation results confirmed the better performance of our algorithm.

In realistic environments, the assumption of *UDG* may be violated, which may cause the idealized planarization algorithms not to work correctly. As a result, *GFG* geographic routing will not work correctly. In the dissertation, we proposed a realistic planarization algorithm in the *Pre-Processed Cross Link Detection Protocol, PPCLDP*, which extracts an almost planar graph, G_{PPCLDP}, from a network graph under realistic environment with obstacles. We proved that *GFG* geographic routing never fails on graph G_{PPCLDP}.

The proposed *PPCLDP* contains a *2-hop Cross Link Pre-Processing (CLPP)* algorithm and a *Restricted Cross Link Detection Protocol (RCLDP)*. In the *CLPP* algorithm, a node detects any *2-hop cross links* of a link attached to it and decides whether to keep or remove the link by exchanging a few messages with its neighbors. The *CLPP* algorithm

generates a graph, G_{CLPP}, which is the input graph of the *RCLDP* and is sparser than the input graph of the *CLDP*. This results in significantly fewer probing messages needed for *RCLDP* than *CLDP*. Our simulation results confirmed that the significantly reduced number of broadcast messages causes *PPCLDP* to have a much lower communication cost and faster convergence time than *CLDP*.

In Greedy-Face-Greedy routing protocols, greedy routing may take packets to concave nodes, where the perimeter routing is applied. This may cause extra hops of routing in networks containing voids. In this dissertation, we proposed a *Hill-Area-Restricted (HAR)* geographic routing protocol. It avoids the extra hops as in the original GFG routing, which makes it more efficient in hop count. We proved that the proposed *HAR* guarantees the packet delivery. Compared to the previous work of *NEAR*, the proposed *HAR* lowers the communication cost tremendously. This makes the *HAR* more desirable for mobile ad hoc and sensor networks. Simulation results showed the better performance of HAR.

5.2 Directions for Future Research

Widely used wireless sensor networks (WSN) contain a large number of embedded devices (sensors) that can collect data, aggregate data and answer queries of data. In WSN, because sensors have limited power due to limited battery life, how to effectively store the large amount of data gathered by sensors, which can increase the efficiency of later data retrieval, is an important issue. In sensor networks, the content of data is more important than the identity of the node that collects the data, which makes data-centric storage [7, 47, 53] widely accepted for data storage.

Ratnasamy et al. [53] proposed a *data-centric storage (DCS)* with *GHT*, a geographic hash table system. In the *GHT* system, the high-level data name, the key, is hashed using a geographic hash table to a geographic location. The sensor node, which is on the geographic location, will store the key-value pair.

When there are voids or obstacles in the network, it is very possible that there is no

node on the geographic location. In this case, *GHT* uses *GPSR* as the routing protocol to route a packet to the appropriate home node nearest to the geographic location. When a packet arrives at its home node, it will enter the perimeter mode for there is no node closer to the destination than the home node. It will traverse the home perimeter until it returns to the home node, which now knows it is the home node and should consume the packet.

The problem is that if the hashed geographic location of a data packet is outside the outer boundary of the network, the packet will traverse the outer boundary that is its home perimeter and be consumed by its home node. In such a case, the operation delay is very long, which means the user has to spend a long time to subscribe or search for a file, which is an inconvenience. In addition, some data stored by a designated home node that is frequently accessed makes all queries for data be routed directly to the home node. This makes the home node a hot spot and cause extra load on the node.

However, given the significant advantages over existing protocols, with further research to optimize the process and eliminate minor problems such as these, *GHT* holds great promise.

Bibliography

[1] K. Akkaya and M. Younis. A survey of routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[2] J.N. AL-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE wireless communications*, 11(6):6–28, Dec. 2004.

[3] S. Ansari, L. Narayanan, and J. Opatrny. A generalization of the face routing algorithm to a class of non-planar networks. In *Mobiquitous*, 2005.

[4] N. Arad and Y. Shavitt. Minimizing recovery state in geographic ad hoc routing. *IEEE Transactions on Mobile Computing*, 8(2):203–217, Feb. 2009.

[5] F. Araujo and L. Rodrigues. Fast localized delaunay triangulation. In *OPODIS2004*, December 2004.

[6] L. Barriere, P. Fraigniaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *DIALM*, Jul. 2001.

[7] F. Bian, R. Govindan, S. Schenker, and X. Li. Using hierarchical location names for scalable routing and rendezvous in wireless sensor networks. In *SenSys 2004*, pages 305–306, 2004.

[8] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications*, 1976.

[9] P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opatrny, G. Salazar, and J. Urrutia. Morelia test: improving the efficiency of the gabriel test and face routing in ad-hoc networks. In *SIROCCO*, pages 23–34, 2004.

[10] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *3rd int. Workshop on Discrete Algorithms and methods for mobile computing and communications*, 1999.

[11] J. Broch, D.A. Maltz, D.B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MOBICOM'98*, pages 85–97, 1998.

[12] S. Capkun, M. Hamdi, and J. Hubaux. Gps-free positioning in mobile ad hoc networks. In *Hawaii International Conference on System Science (HICSS)*, Jan. 2001.

[13] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local construction of planar spanners in unit disk graphs with irregular transmission ranges. In *LATIN*, 2006.

[14] D.P. Dobkin, S.J. Fredman, and K.J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Computational Geometry*, 1990.

[15] R.A. Dwyer. A faster divide-and-conquer algorithm for constructing delaunay triangulations. *Algorithmica*, 2:137–151, 1987.

[16] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. *ISI res. rep. ISU/RR-87-180*, Mar. 1987.

[17] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

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

[19] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanners for routing in mobile networks. *IEEE journal on selected areas in communications*, 23, 2005.

[20] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: a taxonomy. *Ad Hoc Wireless Networking*, 2003.

[21] Venkata C. Giruka and Mukesh Singhal. A self-healing on-demand geographic path-based routing protocolx for mobile ad-hoc networks. *Ad Hoc Networks*, 5:1113–1128, September 2007.

[22] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4:75–123, 1985.

[23] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, Aug. 2001.

[24] L. Hu. Topology control for multihop packet radio networks. *IEEE Trans. Comm.*, 41, 1993.

[25] J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete Computational Geometry*, 7:13–28, 1992.

[26] Neha Jain, Dilip K. Madathil, and Dharma P. Agrawal. MidHopRoute: a multiple path routing framework for load balancing with service differentiation in wireless sensor networks. *Special Issue on Wireless Sensor Networks of the International Journal of Ad Hoc and Ubiquitous Computing*, 1:210–221, 2006.

[27] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353, 1996.

[28] B. Karp. Geographic routing for wireless networks. In *PhD thesis*, 2000.

[29] B. Karp. Challenges in geographic routing: sparse networks, obstacles, and traffic provisioning. In *DIMACS Workshop on Pervasive Networking*, May 2001.

[30] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[31] Y. Kim, R. Govindan, B. Karp, and S. Shenker. Practical and robust geographic routing in wireless networks. In *the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[32] Y. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *USENIX NSDI*, 2005.

[33] Y. Kim, R. Govindan, B. Karp, and S. Shenker. On the pitfalls of geographic face routing. In *ACM DIALM-POMC*, Sep. 2005.

[34] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *11th Canadian Conference on Computational Geometry: an Introduction*, pages 51–54, 1999.

[35] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: of theory and practice. In *ACM PODC 2003*, Boston, MA, USA, July 2003.

[36] F. Kuhn, R. Wattenhofer, and A. 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.

[37] B. Leong, B. Liskov, and R. Morris. Geographic routing without planarization. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation*, volume 3, pages 25–25. USENIX Association, May 2006.

[38] M. Li, W. Lee, and A. Sivasubramaniam. Efficient peer-to-peer information sharing over mobile ad hoc networks. In *the 2nd Workshop on Emerging Applications for Wireless and Mobile Access*, 2004.

[39] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *ACM SenSys'03*, Nov. 2003.

[40] X.-Y. Li, P.-J. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *IEEE ICCCN01*, pages 564–567, 2001.

[41] X.-Y. Li, P.-J. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. In *IEEE HICSS*, 2002.

[42] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *INFOCOM'02*, 2002.

[43] Xu Lin and Ivan Stojmenovic. GPS based distributed routing algorithms for wireless networks. 2000.

[44] C. Liu and J. Wu. Destination-region-based local minimum aware geometric routing. In *Proc. of the 4th IEEE International conference on mobile ad hoc and sensor systems*, pages 1–9, October 2007.

[45] M. Mauve and J. Widmer. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, pages 30–39, Nov. 2001.

[46] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, pages 183–197, Oct. 1996.

[47] J. Newsome and D. Song. GEM: graph embedding for routing and data centric storage in sensor networks without geographic information. In *SenSys 2003*, pages 76–88, 2003.

[48] C. Perkins. Ad-hoc on-demand distance vector routing. In *MILCOM'97*, November 1997.

[49] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing. In *ACM SIGCOMM*, October 1994.

[50] F. P. Preparata and M. I. Shamos. Computational geometry: an introduction. *Springer-Verlag*, 1985.

[51] S. Ramanathan and M. Steenstrup. A survey of routing techniques for mobile communication networks. *ACM/Baltzer Mobile Networks and Applications*, pages 89–104, 1996.

[52] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8:427–442, 2003.

[53] S. Ratnasamy, L. Yin, F. Yu, D. Estrin, R. Govindan, B. Karp, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *WSNA*, pages 78–87, 2002.

[54] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Comm.*, Apr. 1999.

[55] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *ACM MobiCom'06*, Sep. 2006.

[56] K. Seada, A. Helmy, and R. Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *IEEE IPSN Workshop*, Apr. 2004.

[57] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. In *ACM SIGCOMM HotNets*, 2002.

[58] I. Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, pages 2–8, July 2002.

[59] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE transactions on parallel and distributed systems*, 12(10), Oct. 2001.

[60] Yan Sun, Qiangfeng Jiang, and Mukesh Singhal. An edge constrained localized delaunay graph for geographic routing in mobile ad hoc and sensor networks. *IEEE Transactions on Mobile Computing*, 9:479–490, April 2010.

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

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

[63] W. Wang, X.-Y. Li, K. Moaveninejad, Y. Wang, and W.-Z. Song. The spanning ratios of beta-skeleton. In *CCCG'03*, 2003.

[64] Yongwei Wang, Venkata C. Giruka, and Mukesh Singhal. A truthful geographical forwarding algorithm for ad hoc networks with selfish nodes. *International Journal on Networks Security*.

[65] Yu Wang and Xiang-Yang Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications*, 11:161–175, 2006.

[66] Yun Wang, Xiaodong Wang, Demin wang, and Dharma P. Agrawal. Localization algorithm using expected hop progress in wireless sensor networks. In *The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, October 2006.

[67] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for wireless multihop ad-hoc networks. In *IEEE INFOCOM'01*, 2001.

[68] A.C.-C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Computing*, 11:721–736, 1982.

[69] L. Zou, M. Lu, and Z. Xiong. A distributed algorithm for the dead end problem of location based routing in sensor networks. *IEEE Transactions on Vehicular Technology*, 54, July 2005.

[70] L. Zou, M. Lu, and Z. Xiong. A distributed algorithm for the dead end problem of location based routing in sensor networks. *IEEE Transactions on Vehicular Technology*, 54(4), July 2005.

Vita

Yan Sun
Date of Birth: 07/14/1975
Place of Birth: Liaoning Province, China

- **EDUCATION**
  - B.S. in Computer Science, May 1997. Tongji University, Shanghai, P.R. China.
- **PROFESSIONAL SKILLS**
  - Programming languages: Visual Basic, C, mysql, perl, php, html
  - database development, web page development, data-driven web design
  - Operating Systems: Unix, Linux, Windows XP/NT/9x
  - Software: Microsoft Office, Visual Studio
- **RESEARCH INTERESTS**
  - **Routing in Mobile Ad Hoc and Sensor Networks:** Geographic routing in mobile ad hoc and sensor networks.
  - **Data-Centric Storage in sensor networks:** Data-Centric storage methods that increase the efficiency of data retrieval in sensor networks.
  - **Security on Mobile Peer-to-Peer Systems:** Security on the Gnutella and other mobile P2P systems.

- **PUBLICATIONS**
  - **Journal papers:**
    - **Yan Sun**, Q. Jiang and M. Singhal. "An Edge Constrained Localized Delaunay Graph for Geographic Routing in Mobile Ad Hoc and Sensor Networks". *IEEE Transaction on Mobile Computing* 2010 volume 9, number 4, pages: 479-490.
    - **Yan Sun**, Q. Jiang and M. Singhal. "A Pre-Processed Cross Link Detection Protocol for Geographic Routing in Mobile Ad Hoc and Sensor Networks under Realistic Environments with Obstacles". *Journal of Parallel and Distributed Computing* 2011 volume 71, issue 7, pages: 1047-1054.
  - **Conference publications:**
    - **Yan Sun**, Q. Jiang and M. Singhal. "An Edge Constrained Localized Delaunay Graph for Geographic Routing in Mobile Ad Hoc Networks". In Proceedings of Wireless Communications and Networking Conference 2007 (WCNC 2007), Hong Kong, P. R. China, March 11-15, 2007, pages 4002-4007.
- **PROFESSIONAL ACTIVITIES**
  - A reviewer for IEEE Globecom, IEEE Transactions on Parallel and Distributed Systems (TPDS), and International Symposium on Reliable Distributed Systems (SRDS).
- **TEACHING INTERESTS**
  - data structures, discrete mathematics and algorithm
  - database development
  - programming languages, i.e., Visual Basic, C, mysql, perl, php
  - program design and development
  - software engineering
  - web page development and data-driven web design
- **PROJECTS**
  - Simulator: Developed a simulator written in C, by which nodes in the network can communicate with each other by sending and receiving messages.
  - Data-Driven Web Design: Design a data-driven web page written in perl.
- **REFERENCES**
  - Dr. Mukesh Singhal, Professor and Gartner Group Endowed Chair in Networking
    Department of Computer Science, University of Kentucky
    234 James F. Hardymon Building, 301 Rose Street
    Lexington, KY 40506-0495

Phone: (859) 257-3062, Fax: (859) 323-3740
E-mail: singhal@cs.uky.edu

– Dr. D. Manivannan, Associate Professor
Department of Computer Science, University of Kentucky
231 James F. Hardymon Building, 301 Rose Street
Lexington, KY 40506-0495
Phone: (859) 257-9234, Fax: (859) 323-3740
E-Mail: mani@cs.uky.edu

– Dr. Raphael A Finkel
Professor and Director of Graduate Studies
Computer Science Department, University of Kentucky
228 James F. Hardymon Building, 301 Rose Street
Lexington, KY 40506-0495
Phone: (859) 257-3885, Fax: (859) 323-3740
E-Mail: raphael@cs.uky.edu

– Dr. Zongming Fei, Associate Professor
Department of Computer Science, University of Kentucky
227 James F. Hardymon Building, 301 Rose Street
Lexington, KY 40506-0495
Phone: (859) 257-3202, Fax: (859) 323-3740
E-Mail: fei@netlab.uky.edu

– Cai-Cheng Lu, Professor
Department of Electrical and Computer Engineering, University of Kentucky
693 F. Paul Anderson Tower, Lexington, KY 40506-0046
Phone: (859) 257-1644, Fax: (859) 323-3092
E-mail: cclu@engr.uky.edu

Yan Sun