Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Yanli DENG

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Partial Delaunay Triangulations Based Routing, Address Configuration and Data-centric Storage in Ad hoc Networks

I. Stojmenovic

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

A. Boukerche

E. Kranakis

J.-M. De Koninck, Ph.D

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Partial Delaunay Triangulations Based Routing, Address Configuration and Date-Centric Storage in Ad Hoc Network

Yanli Deng

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Electrical Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

April 2004

Your file Votre référence
ISBN: 0-494-01458-X
Our file Notre référence
ISBN: 0-494-01458-X

Acknowledgements

I would like to acknowledge my sincerest gratitude to my supervisor Prof. Dr. Ivan Stojmenovic of Ottawa-Carleton Institute for Computer Science. Without his guidance and enormous help during the development phase of the Simulation program and writing this paper, it would not be possible to complete the thesis. I would also like to thank my family for their continuous support.

Table of Contents

List of Figures

Abstract

The work of this thesis is based on a recently proposed memoryless *Greedy-Face-Greedy* (*GFG*) routing algorithm that guarantees delivery in connected unit disk graphs (where two nodes are connected if and only if their distance is no more than the transmission radius, which is equal for all nodes). The FACE mode is a recovery mode used when no neighbor closer to destination exists. FACE mode requires extracting a planar sub graph out of the unit disk graph. We propose to apply recently proposed *Partial Delaunay triangulation* (PDT) instead of *Gabriel Graph* (GG) used in the original *GFG*. PDT is locally defined without any message exchange in addition to those needed to learn the locations of neighbors. Routing is further enhanced by applying *dominating set* based routing.

We consider two solutions for generating IP addresses: each node generates IP addresses following its own scheme, or all addresses are generated at a unique node in the network, called the *Addressing Agent* (AA). While in previous proposal nodes look for the AA by flooding their requests to, say, a node with the lowest ID, we propose to route request until the easternmost node on the outer boundary is found. We propose to apply a hash function to assigned IDs (whether assigned by nodes themselves or by the AA), and map each ID to a FACE of the PDT. Duplicates will be detected along the FACE. Each ID traverses the assigned FACE, and detects duplicates if there are any. The new address is selected and the ID is sent to the appropriate FACE for the verification. When no duplicate is detected, the message returns to the node with the confirmed or newly selected ID address. Due to hash function selection or node mobility (or if a random direction is used instead of a hash function), the assignment may frequently be made to the outer perimeter of the network, so this special case is considered separately here.

In data storage systems, data is stored in the network conveniently so that it can be easily found by the node that needs it. In recently proposed data storage schemes, the key for the data is hashed into a node in the network and data is routed toward it. It is stored in all nodes along the FACE containing a hashed location. This proposal assumes GG structure for the FACE. We propose to apply PDT instead, which is a denser structure resulting in a smaller perimeter.

Our new schemes are implemented and compared experimentally with competing schemes, showing their advantages.

Chapter 1 Introduction

This chapter discusses about the backgrounds of the thesis, the problem statements, existing solutions, objectives and contributions of the thesis. It gives a general description of the whole paper.

1.1 Background

A mobile ad hoc network (MANET) is a group of mobile, wireless nodes that can communicate with each other without any fixed infrastructure or centralized administration. A node communicates directly with other nodes within its wireless communication range. It can also communicate with nodes that are beyond the wireless ranges by using multi-hop routes through intermediate nodes in the network. These multi-hop routes change with the change of network topology. Each node in an ad hoc network is in charge of routing information among its neighbors, thus contributing to and maintaining connectivity in the network. In ad hoc networks, nodes collaborate with each other in order to transport information. These nodes act as end systems and routers at the same time. Since ad hoc networks have many proven benefits, they are the subjects of much current research. Examples of such networks are *Wireless Local Areas Networks* (*WLAN*), packet radio networks and sensor networks. Possible applications of these networks are in situations like disaster rescues, wireless teleconferences, battlefield, or monitoring objects in a possibly remote or dangerous environment.

In ad hoc networks, some nodes are static (e.g. rooftop networks, sensor networks); some nodes in the networks are mobile (e.g. laptops, vehicle mounted devices). Wireless sensor networks can be used in a remote terrain or a toxic environment. They can greatly extend our ability to monitor and control the physical environment from remote locations and improve the accuracy of information obtained via the collaboration of sensor nodes and on-time information processing at these nodes. A sensor network is a distributed network comprised of a large number of small devices, each with some computational, storage and communication capability. Sensor networks can operate in an unattended mode to record detailed information about their surroundings. They are well suited to applications such as location tracking and habitat monitoring. These sensor networks may scale in size, as will the amount of data they make available. Therefore, making effective use of sensor networks will require scalable and self-organizing data dissemination

algorithms. Design of routing protocols is a crucial problem in ad hoc networks, and many routing algorithms have been developed.

1.2 Problem statements

In a self-organized network, routing is a key issue for the whole network. The task of finding and maintaining routes in a network is nontrivial because a host's status may change from time to time. A host may change its location from one place to another because of its mobility, or it may turn on or off its power to save energy. The task of routing is to send a message from a source to a destination. Due to propagation limitations, the transmission radii are limited.

This paper uses position-based routing algorithms for ad hoc networks. We are interested only in the design of localized routing schemes. By calling the routing scheme "localized" we mean that each node decides its forwarding decision according to its own geographic location, the geographic location of the destination and its 1-hop neighbors' geographic locations. Because of the characteristics of ad hoc networks, most routing protocols in fixed networks don't adapt to ad hoc networks. The function of routing protocols is to find a path between the source and the destination. Unit Disk Graph (UDG) is a good model to represent ad hoc networks. UDG is defined as following [DSW]:

> Let S be the set of points in a plane. Then the unit disk graph $U(S)$ is a geometric graph that contains a vertex for each element of S. An edge (u, v) is present in $U(S)$ if and only if $d(u, v) \leq R$, where $d(u, v)$ denotes the Euclidean distance between u and v; R is the transmission radius of nodes and R is equal to all nodes.

An edge between two nodes on a UDG means these two nodes have duplex communication. So the job to find a route between two nodes in an ad hoc network is to find a connection between these two nodes on a UDG. The delivery rate is defined as the number of received packets divided by the number of sent packets. If the delivery rate equals 1, this routing protocol can guarantee delivery. Assuming an ideal Media Access Control (MAC) layer, we want to design routing schemes that guarantee delivery and minimize hop count for doing so. That is, we want a performance as close to the performance of the *shortest path algorithm* as possible.

Address configuration is a major issue in ad hoc networks. Ad hoc networks don't have fixed *Dynamic Host Configuration Protocol* (DHCP) servers. How to give each node in MANET a unique IP address is a challenging job. A dynamic and self-organized mechanism is needed for address configuration in ad hoc networks. This mechanism should be easy to handle splitting and merging of networks, node joining and node leaving the network. *Duplicate Address Detection* (DAD) is important in ad hoc networks. Duplicate addresses are more likely to happen in ad hoc networks than in fixed networks. An effective approach is needed to settle this problem. This approach should have small communication overheads and be localized.

Sensornets can collect huge amount of information from the environments for human beings. There needs to be an efficient method to store and retrieve the overwhelming amount of data gained from a sensornet. We need a scalable, self-organized data dissemination approach for data storage in ad hoc networks. This approach should minimize communication overheads.

1.3 Existing solutions

Many position-based routing algorithms have been proposed recently. The *greedy routing* scheme [F, TK, SL] chooses its next hop as close to the destination as possible. In *greedy routing* schemes, the node currently holding messages will forward them to neighbors whose positions in terms of distance, progress, or direction is the best among all neighbors. *Greedy routing scheme's* performance is very close to the *shortest path algorithm* on dense networks. But *greedy routing* scheme can't guarantee delivery. The *FACE routing* algorithm [KSU, BMSU] and the *GFG routing* algorithm [BMSU] can guarantee delivery. They only use localized manner to forward packets. But *FACE routing* algorithm and *GFG routing* algorithm take a relatively longer path from source to destination compared with the *shortest path algorithm*. Improvements based on the use of *dominating set* and the shortcut scheme, which require 2-hop local knowledge, are proposed in [DSW]. There is room to further reduce hop count for these two routing schemes. More efficient routing schemes of the *FACE routing* algorithm and the *GFG routing* algorithm need to develop.

The *FACE routing* algorithm uses localized information. Any sending node only uses its own location, the destination location and its immediate 1-hop neighbors' locations to forward packets. Also the *FACE routing* algorithm uses the *right hand rule*. It first extracts a planar

graph from the UDG that is used to represent the ad hoc network. The planar graph is composed of FACEs. Packets travel from one FACE to another, getting closer and closer to their destinations. A GG graph is used to extract the connected planar graph. A GG graph is defined in the following way:

> Let S be a set of points in the plane. The GG graph GG(S) is a geometric graph in which the edge (u, v) is present if and only if disk(u, v) contains no other points of S where disk(u, v) denotes the disk with diameter uv.

The problem with the *FACE routing* algorithm is that the *FACE routing* algorithm adopts a snake-like path. This results in a much longer route. The *GFG routing* algorithm has a much better performance. The *GFG routing* algorithm runs *greedy routing* algorithms as much as possible. When *greedy routing* algorithm fails at a local maximum point, the *GFG routing* algorithm changes to FACE mode to recover from failing. It will change back to greedy mode if the packet encounters a node that is closer to the destination than the local maximum point. The alternation between FACE mode and greedy mode may happen several times before the packet gets to the destination. The *GFG routing* algorithm has a performance that is very close to the *shortest path algorithm* on dense networks. The *GFG routing* algorithm also works on a GG. A GG can be constructed in a localized manner, but it may not be the best choice for the *GFG routing* scheme.

Address configuration is important in ad hoc networks. Unlike traditional networks, ad hoc networks can't use fixed DHCP servers to distribute IP addresses. Any fixed server can be disconnected from the network from time to time in the ad hoc network. An *Addressing Agent* approach is proposed in [GR]. This approach finds a node with the lowest MAC address as the *Addressing Agent*. This approach can easily handle splitting and merger. The problem is that finding the node with the lowest MAC address is not easy. It needs flooding within the whole network. This will need a lot of communication overheads. There is also the possibility that two nodes may have the same MAC address. Duplicate address detecting may be difficult because it needs the participation of all nodes in the network.

When two ad hoc networks merge, there is the possibility that two nodes have the same IP address. Some proposals use broadcast mechanisms to avoid duplicate addresses [OMG, NP]. In these mechanisms, if a node wants to use an IP address, it broadcasts an inquiry to the whole

network. If any node already uses this IP address, it will reply to the inquiry through this node. This broadcast mechanism will generate many communication overheads in the network.

Data-Centric Storage (DCS) is a good method for data storage in ad hoc networks. DCS uses the *Geographic Hash Table* (GHT) to determine a position for data storage. DCS first constructs a planar graph, and then it uses a GG graph to extract a planar graph from the sensornet [KSEGYY]. The GG planar graph has many FACEs. A perimeter is the boundary of a FACE on a planar graph. The outer perimeter is the outer boundary of a planar graph. The hashed points locate within the perimeters. A perimeter where a hashed point is inside of it is called the *home perimeter* for this point. The *GFG routing* algorithm is used to locate the hashed point. The hashed point can be seen as a disconnected node. When using the *GFG routing* scheme packets will travel along the perimeter that encloses the hashed point. The nearest node to the hashed point is called the *home node*. A *Home node* is used to store date. All other nodes on the *home perimeter* are called replicas. Replicas copy date from the *home node* in case the *home node* fails. In DCS, the communication overheads mostly happen on *home perimeters*. The size of a *home perimeter* will affect the performance of DCS. In [KSEGYY], a GG is used to construct a planar graph.

1.4 Thesis objectives and contributions

One objective of this thesis is to develop more efficient routing schemes to reduce average hop count. The routing scheme will have the following performance:

- Guaranteed delivery
- Localized manner
- Minimized average hop count
- Single path strategy

In our solutions, we use a localized geometric structure called PDT, which is defined as follows.

> A Delaunay triangulation (DT) consists of triangles (with vertices from node set) whose circumscribes do not contain other nodes inside them. PDT [LSW] is a portion of Delaunay triangulation consisting of edges that can be confirmed locally as being part of DT, based

on 1-hop local information, without any messages being exchanged between neighboring nodes.

The proposed routing scheme uses PDT to extract a planar graph from UDG that is used to model the ad hoc network. A PDT graph is denser than a GG graph. The *GFG routing* algorithm has a better performance on a PDT than on a GG. PDT based *GFG routing* has a hop count which is about 8.5% lower than GG based *GFG routing* on sparse networks with N=60 and d=4. Here N is the number of total nodes and d is the average degree of a UDG. PDT based *GFG routing* has a hop count which is 6.5% lower than GG based *GFG routing* on dense networks with N=60 and d=15. The average hop count becomes smaller as d increases.

We incorporate *connected dominating sets* (CDS) with the proposed routing schemes. More specifically, we apply the *inter-gateway node* set definition [WL, SSZ], as in [DSW]. A node is a *inter-gateway node* if it is not covered by any neighboring node with a higher ID value, and has two unconnected neighbors [WL, SSZ]. Node A is covered by node B if any neighbor of A is also a neighbor of B. PDT based *GFG routing* incorporated with the *inter-gateway* scheme has an obvious effect on sparse networks. It reduces hop count by 20% on sparse networks with N=60 and d=4 compared with routing that does not incorporate the *inter-gateway* scheme. The reduction of average hop becomes smaller as d increases. The shortcut scheme [DSW] is also used in the routing algorithm if nodes have 2-hop neighbor information. In the shortcut method, a node A currently holding a message uses its 2-hop knowledge to predict several forwarding moves for the message, as long as it has sufficient knowledge to decide it, and as long as the next node on the path is its neighbor. Node A then sends a message directly to the last node on the FACE route that is still its neighbor, instead of strictly following the *FACE routing* scheme hop by hop. PDT based *GFG routing* incorporated with the shortcut scheme also has good effects on sparse networks. It reduces hop count by 26.9% on sparse networks where N=60 and d=4 compared with routing that does not incorporate the shortcut scheme. The reduction of average hop becomes smaller as d increases

We propose a new algorithm to generate random connected unit disk graphs. We first find the transmission radius that corresponds to the desired average degree k, by assuming that the number of neighbors of each node is equal to the ratio of the area of a circle with given transmission radius and the area of a square where the nodes are located, multiplied by the number of other nodes. Each node generates its coordinates at random, but is only accepted if it

is the neighbor of at least one other already generated node. A connected unit disk graph is generated at the end, but its exact average degree, d, is different from the desired one, k. The transmission radius (larger or smaller) is then corrected so that the average degree becomes k. Dijkstra's *shortest path algorithm* is then applied to test connectivity. This algorithm results in fast graph generation for sparse graphs, which is the bottleneck in the previous generation method.

The task of address configuration in ad hoc networks is to assign and maintain a unique IP address at each node. There are several proposed solutions in the literature on the subject, however, none of them properly address all the issues, i.e., generating new IP addresses, merging two networks, detecting duplicate addresses, and re-assigning new addresses to these duplicates.

We will consider two solutions for generating an IP addresses: where each node generates an IP address following its own scheme, or where all addresses are generated at a unique node in the network called the *Addressing Agent* (AA). While in previous proposal nodes look for the AA by flooding their requests to, say, a node with the lowest ID, we propose to route a request until the easternmost node on the outer perimeter is found. We propose to apply a hash function to assigned IDs (whether assigned by a node itself or by the AA), and map each ID to a FACE of the PDT. Duplicates will be detected along the FACE. Each ID traverses the assigned FACE, and detects duplicates, if any. A new address is selected and the ID is sent to the appropriate FACE for verification. When no duplicate is detected, the message returns to the node with the confirmed or newly selected ID address. Due to the hash function selection or node mobility (or if a random direction is used instead of the hash function), the assignment may frequently be made to the outer perimeter of the network. This is special case and we will consider it separately here.

This paper proposes to use a PDT to replace a GG in the DCS. A better planar graph PDT could improve the performance of the DCS. A PDT can be constructed in a localized manner and has a smaller average size of perimeters. This property can reduce the average size of the *home perimeters* and thus the local communication overheads are reduced.

In ad hoc networks, routing, address configuration and data-centric storage are strong connected. In this thesis, *GFG routing* is used to find *home perimeters* and *home nodes* for address configuration. Data storage and retrieval in data-centric storage also use *GFG routing* to locate

home perimeters and *home nodes*. Add configuration provides IP addresses for the formation of ad hoc networks. Both address configuration and data-centric storage use PDT planar graph to construct *home perimeters*. In general, these three protocols are relevant with each other.

1.5 Conditions, assumptions and limitations

The following assumptions are used in this thesis:

- Each node exactly knows the locations of all of its 1-hop neighbors.
- The network topology remains the same as we find all paths between any pair of nodes.
- Every node has the same wireless transmission range.
- There are no obstacles between any two nodes.
- The media access layer is collision-free.
- Although a source node may not know the locations of intermediate nodes, it will know the location of the destination node when it sends packages.
- For *dominating set*s and shortcut schemes, nodes are assumed to have 2-hop neighbor information.
- All nodes know their own locations.

1.6 Organizations

This paper is organized as following: Chapter 2 gives a literature survey; Chapter 3 talks about PDT based routing protocols and shows experiment results; Chapter 4 describes PDT-perimeter based address configuration scheme and DAD; Chapter 5 describes data-centric storage in sensornets; Chapter 6 discusses conclusions and future researches.

Chapter 2. Literature review

Since the advent of DARPA packet radio networks in the early 1970's, numerous routing protocols have been developed for ad hoc networks. These protocols must deal with the typical properties of these networks.

One of the fundamental problems in routing algorithms is the shortest path problem. Briefly, given a graph G in which there is a cost (weight) associated with each edge, the goal is to compute the least expensive path between pairs of vertices of G. These routing protocols may generally be categorized as topology-based routing protocols and position-based routing protocols. In topology-based routing protocols, each node has knowledge of part of or the whole network. For example, a node may know about its neighbors (1-hop knowledge), and all neighbors of each neighbor (2-hop knowledge). In such scenarios, destinations are found by flooding or by using the Bellman-Ford type of routing protocols (that is, routing tables, as on the Internet). Some protocols assume that each node knows the full graph topology, that is, all nodes and all edges in the graph (global knowledge). In this case, a node can apply a *shortest path algorithm* to find a route to any node. In contrary, position-based routing protocols only use location information. They don't have the communication overheads needed to maintain routing tables. This thesis focuses on position-based routing protocols.

This chapter discusses about the concept of the UDG that is used to model an ad hoc network, different ways to construct planar graphs, and different existing routing protocols for ad hoc networks. It summarizes the advantages and drawbacks of the mentioned routing protocols. This chapter also discusses some existing protocols for address configuration and data-centric storage.

2.1 Unit disk graphs

a UDG is a ideal graph-theoretical model for wireless networks. A UDG can be constructed in the follow way:

> Each node A has its transmission range $t(A)$ thus joined by an edge if the Euclidean distance between their coordinates in the network are less than the minimum between their

transmission radii (i.e. $d(A,B) < \min\{t(A), t(B)\}$). If all transmission ranges are equal, the corresponding graph is a UDG.

UDG models provide acknowledges for received messages. UDG models are valid models when there are no obstacles on the signal paths (e.g. a building). If there are obstacles within the ad hoc network, the ad hoc network can be modeled by a sub graph of a UDG.

There are two important parameters for a UDG: total node number N and average degree d. These two key parameters decide many important properties of a UDG. The degree of a node represents the total number of connections this node has. The average degree of a UDG is the sum of every node's degree divided by the number of total nodes, N. Hereafter we use d to refer to the average degree of a UDG.

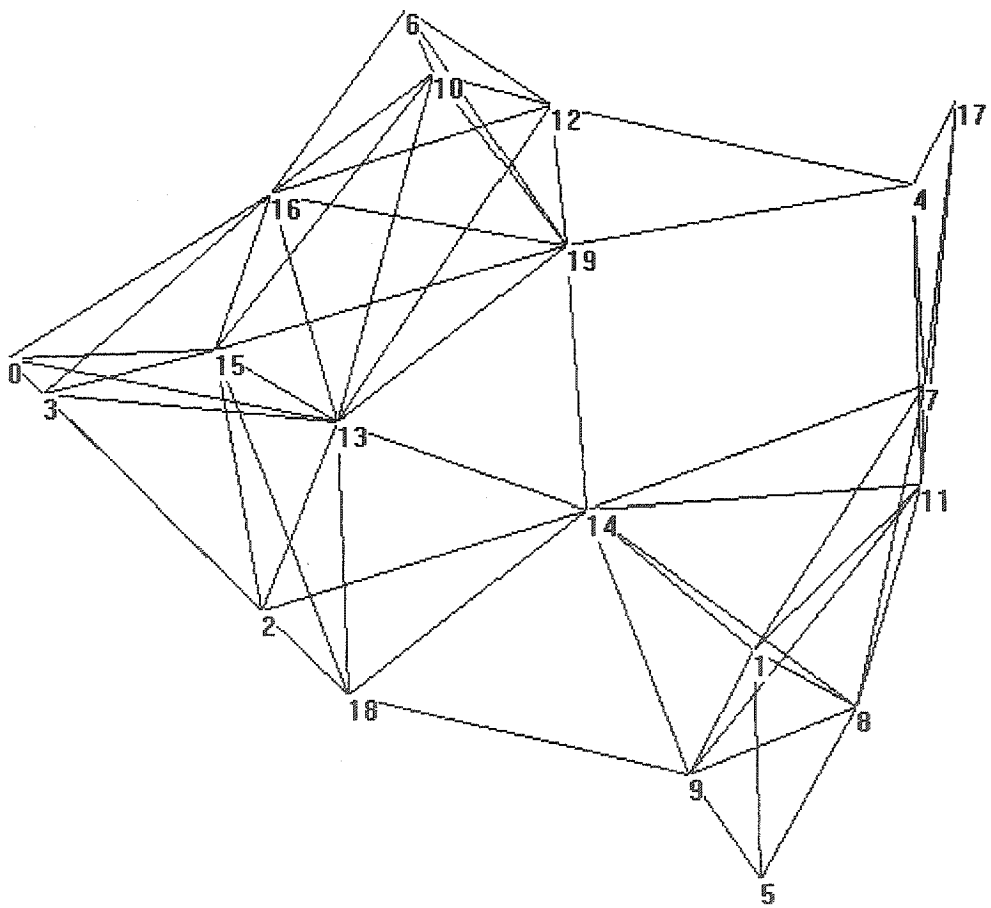


Figure 1. An example of a UDG with N=20 and average degree d=6

The following method is used to generate a required random UDG with expected average degree d and the number of total nodes N: Each node randomly obtains its x coordinate and y coordinate in an interval [0, m). Then all $N*(N-1)/2$ potential edges in the network among the N nodes are

sorted by their length in ascending order. The radius R that corresponds to a chosen value of d is equal to the length of $N*d/2$-th edge in the sorted order. Any edges smaller than R will stay in the graph. Others are eliminated from the graph. Dijkstra's *shortest path algorithm* (from one node to all other nodes) is used to check the connectivity of the graph. A cost is used to weight an edge. If there is an edge between two nodes, we set the cost between the two nodes to be 1; if there isn't an edge between two nodes, we set the cost between the two nodes to be infinite. (In the program, a big enough value is used to express infinite). If the graph is connected, any cost from one node to any other node should not be infinite; otherwise, the graph is not connected. If an unconnected graph is found, this graph is not valid and it is ignored, we repeat above procedures until a qualified graph is obtained.

This simple approach generates a full random UDG. It is useful to generate a UDG with a relatively large degree. It may take a very long time to find a qualified UDG with small degree because of the low probability of such UDG.

2.2 Connected planar graph

2.2.1 Gabriel graph (GG)

A GG graph is a planar graph. The following text is the definition of a GG.

Definition: Let disk(u, v) be the disk with the diameter (u, v), then edge uv belongs to Gabriel graph GG(S) if and only if disk(u, v) contains no other nodes from S. Figure 11 shows an example of an edge in a GG.

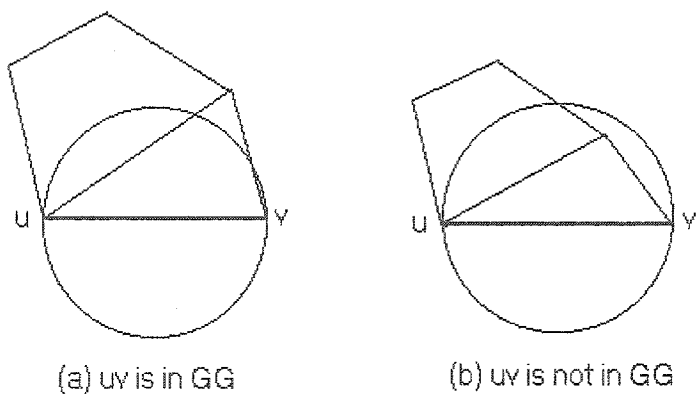


Figure 2 A scenario of edge in a GG

According to the definition of a GG, we can use the following way to construct a GG. Let m be the middle point of edge uv. The length of edge uv is $|uv|$. For any node x in the UDG, the edge uv is in a GG if and only if the length $|xm|>|uv|/2$. Figure 3 shows the extracted GG from the UDG.

It has been proven that a GG is planar and connected [BMSU]. A GG can be constructed in a localized manner. Whether an edge belongs to a GG or not can be determined by one endpoint of this edge. The node only needs to know the locations of its 1-hop neighbors. The following is the pseude-code to generate a GG

For all v

GG*edge(v)=true;*

For all w

If v<>w and d(u,m)> d(m,w) then GG*edge(v)=false;*

End for

End for

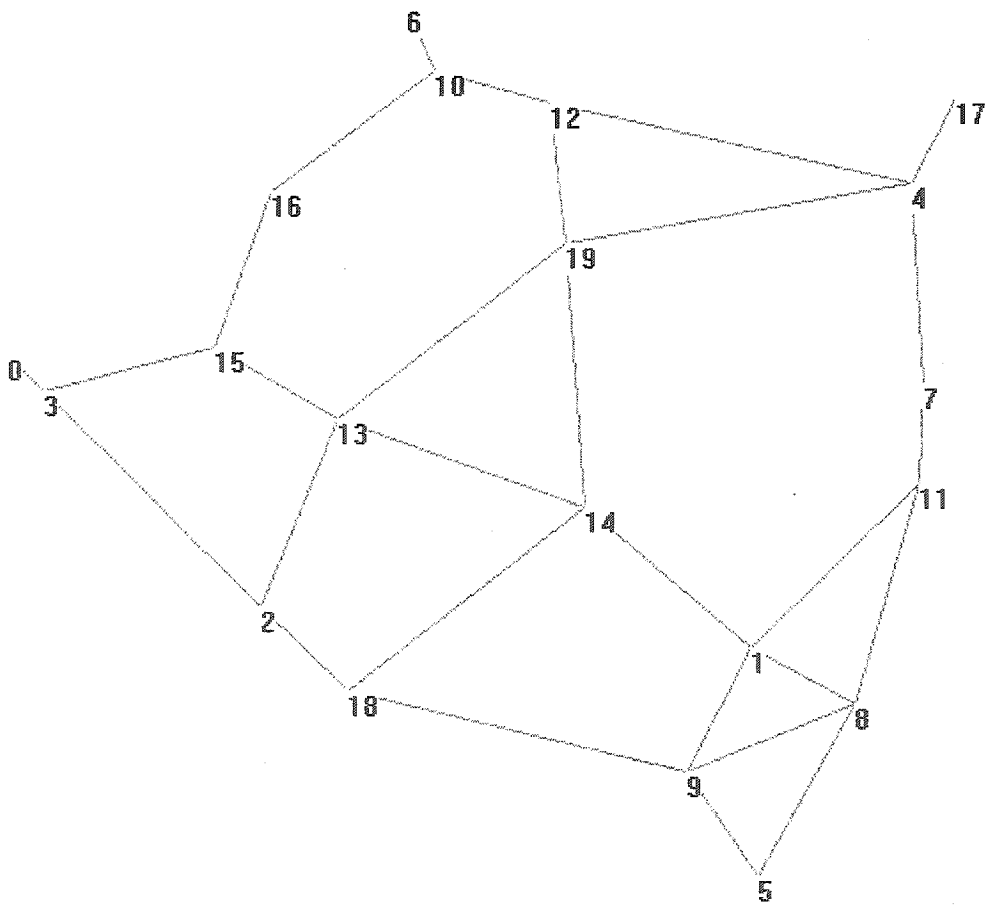


Figure 3 A GG extracted from the UDG

2.2.2 PDT graph

To discuss a PDT graph, we first describe a DT graph.

a) DT graph

Definition: Edge *uv* is in UDG. *S* is the set of all nodes. Edge *uv* belongs to a DT graph Del(*S*) if and only if there exists a circle with *u* and *v* on its boundary which doesn't contain any other node from the set *S*. The DT graph contains all the edges that satisfy the conditions.

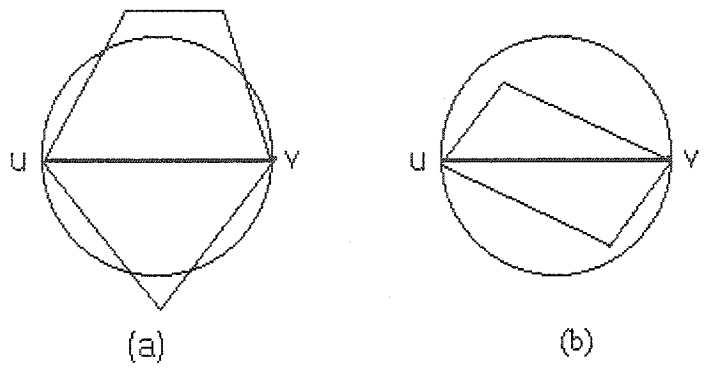


Figure 4 A scenario of a DT graph: *uv* in a) is in a DT, in b) not in a DT

The procedures to determine whether an edge *uv* belongs to Del (*S*) include following steps:

Algorithm: constructing GG

Input (*UDG*)

Output (*DT*)

First check whether the disk(*u, v*) contains any other nodes from the network. If it doesn't, then edge *uv* belongs to Del(*S*).

If the disk(*u, v*) does contain nodes from the network, check whether these nodes exist on both sides of edge *uv* or only on one side of edge *uv*. If both sides of edge *uv* have nodes from the network inside the disk(*u, v*), then the edge *uv* does not belong to Del(*S*). In this case, there will not exist a circle with nodes *u* and *v* on its boundary that does not contain any other nodes from the set *S*.

If only one side of edge uv has nodes inside the disk(u, v), we assume w to be one such node that has the maximum angle $\angle uwv$. We draw a circle passing through nodes u, w and v. If the circle doesn't contain any other nodes from the network, then edge uv belongs to Del(S). Otherwise, edge uv doesn't belong to Del(S).

The above method is the standard procedures to check whether an edge belongs to a DT graph. But this procedures are not good for computing. There is a simpler way to construct a DT: Let w be the node with the biggest angle $\angle uwv$ on one side of edge uv and let x be the node with the biggest angle $\angle uxv$ on another side of edge uv (If there is no such a node x, let $\angle uxv = 0$). If $\angle uxv + \angle uwv \geq \pi$, then edge uv is not in the DT graph Del(S). Edge uv belongs to Del(S) if and only if $\angle uxv + \angle uwv < \pi$.

It has been proven that a DT graph is planar and connected [LSW]. A DT graph has more edges than a GG graph. Figure 5 shows the DT graph extracted from the UDG.

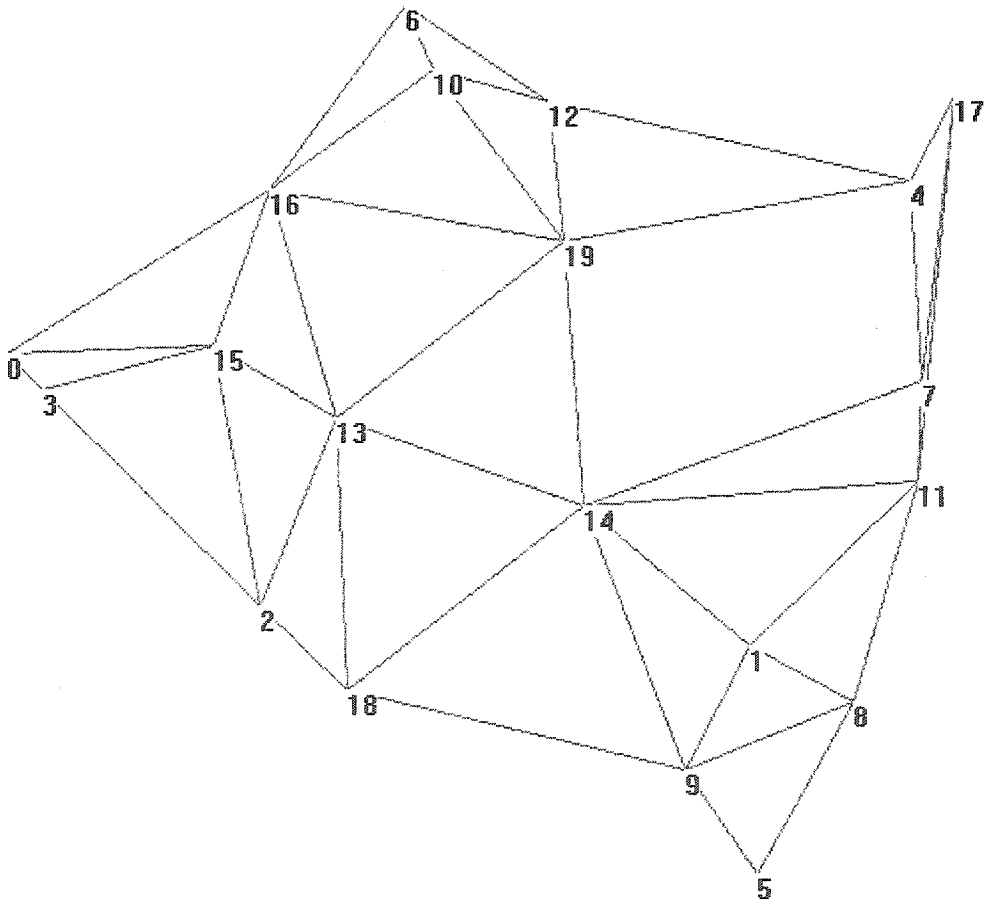


Figure 5 A DT graph extracted from the UDG

b) PDT graph

Li, Calinescu, Wan and Wang [LCWW] proposed a LDT (Localized Delaunay Triangulation) structure to replace the GG. Their structure contains all the edges that are common to Delaunay triangulation and a unit disk graph, and some more. However, their structure requires message exchange between neighboring nodes to construct it, in addition to messages needed to learn about the position of neighbors. The maintenance of the structure also involves additional messages. We propose to use the structure that is also localized, and requires no message exchange to construct and maintain the structure (PDT), in addition to messages needed to maintain neighborhood information. A PDT is a subset of a LDT and therefore a LDT is denser and should reduce more hop count on routing; however this is at the cost of additional setup messages. To overcome the drawbacks of the DT and the LDT, we use a PDT for a planar graph. A PDT graph contains a GG as its sub graph, and a PDT itself is a sub graph of a DT graph. The definition for construction of a PDT graph goes as follows:

Definition: Let $N_1(u)$ denote all 1-hop neighbors to the node u. Let w be the node from the set $N_1(u)$ that is on one side of edge uv with the largest angle $\angle uwv$. Edge uv is added to a PDT if the following conditions hold: (1) there is no node from $N_1(u)$ that lies on the different sides of uv with w and inside the circumcircle passing through u, v and w. (2) $\sin\alpha > d/R$, where R is the transmission radius of each node and $\alpha = \angle uwv$ (here $\alpha < \pi$). Figure 6 shows a PDT extracted from the UDG.

From the definition, we can see that node u can determine whether edge uv belongs to the PDT only by its local information. Because the status of edge uv is only dependent on node u, a question arises as to whether or not the status of uv, as decided by u, will contract the status of uv as decided by v. Thus we arrive at the following Lemma.

Lemma: There are two nodes, u and v, in a UDG and edge uv exists in the UDG. If edge uv belongs to a PDT graph from the view of u, then uv will also belong to the PDT graph from the view of v.

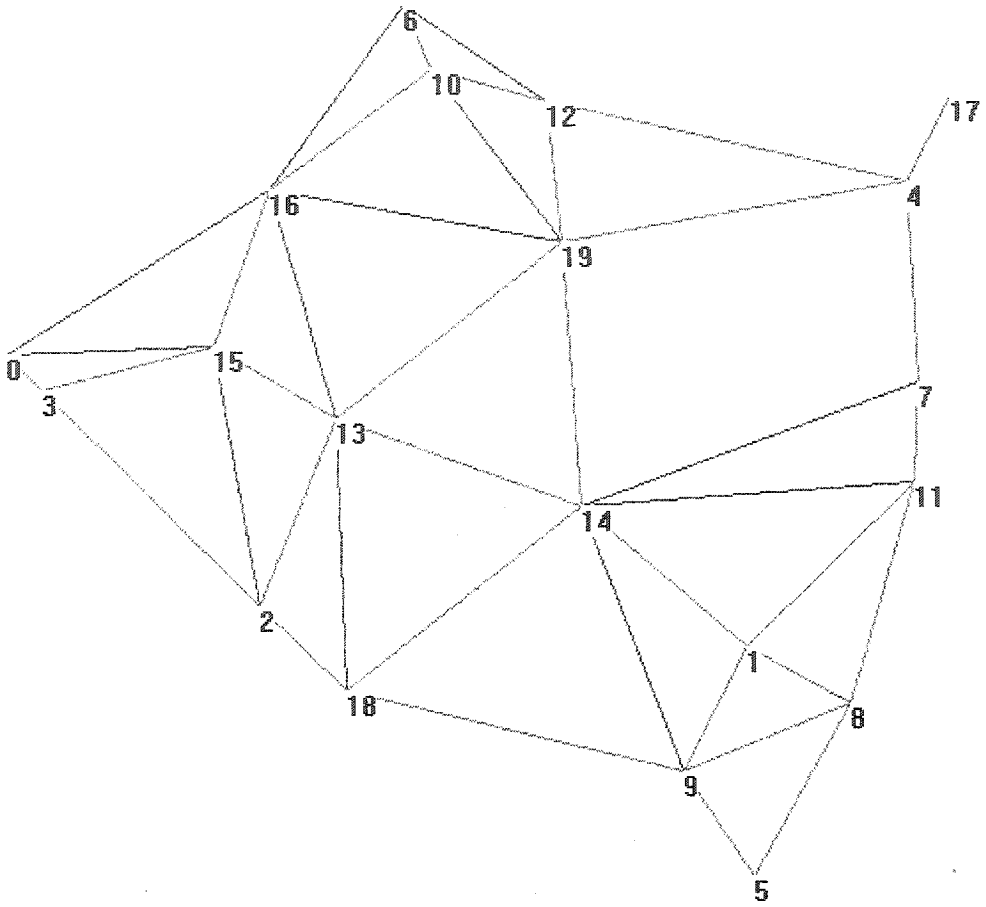


Figure 6 A PDT graph (Blue and pink belong to the PDT; pink belongs to the GG)

This lemma is important for drawing a PDT graph in a localized manner. A PDT graph drawn in a localized manner will not conflict with the global architecture. The PDT graph constructed by the definition is only valid for a UDG. It is not useful for arbitrary graphs. To draw a PDT graph from a UDG, we can judge whether an edge is in the PDT graph only from one endpoint of an edge.

2.3 Topology-based routing schemes

Topology-based routing schemes include proactive routing schemes and reactive routing schemes. Proactive schemes employ classical routing strategies. The main drawback of these approaches is that the maintenance of unused paths may occupy a significant part of the available bandwidth if the topology of the network changes frequently. Proactive algorithms include the *Destination-Sequence Distance Vector* (DSDV) [PB] and the *Wireless Routing Protocol* (WRP) [MG].

A different approach from proactive routing is the source-initiated on-demand routing scheme. This type of routing scheme finds routes only when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, some forms of route-maintenance procedures maintain it until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

Some popular source-initiated on-demand routing protocols include: *Ad hoc On-Demand Distance-Vector* (AODV) [PR], *Dynamic Source Routing* (DSR) [JM], *Location Aided Routing* (LAR) [KV] and *Temporally-Ordered Routing Algorithm* (TORA) [PC]. Topology-based routing protocols are not the topic of this thesis.

2.4 Greedy position-based routing scheme

Greedy position-based routing schemes eliminate some of the limitations of topology-based routing schemes by using additional location information. They require that information about the physical positions of the participating nodes be available. Commonly, each node determines its own position through the use of the *Global Position System* (GPS) or some other type of positioning service. The routing decision at each node is then based on the destination's position contained in the packet and the positions of the forwarding node's neighbors. Thus position-based routing schemes don't require the establishment or maintenance of routes. The nodes neither have to store routing tables nor transmit messages to keep routing tables up to date. The distance between any two adjacent nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors. The location of nodes may be available directly by receiving signals from GPS satellites if nodes have the small, inexpensive low-power GPS receivers.

Most Forward within Radius (MFR) is the first position-based routing scheme based on the notion of the progress [TK]. The progress of a node is defined as the projection onto the line connection S and D. S is the current sending node and D is the final destination. MFR chooses the next hop that has the biggest progress. MFR routing scheme tries to minimize the number of hops a packet has to traverse in order to reach D. MFR has proven to be a loop-free scheme [SL]. MFR routing scheme is a competitive progress-based algorithm in term of hop count.

Nearest with Forward progress (NFP) is another routing scheme using the notion of progress [HL]. In NFP, the packet is transmitted to the neighbor nearest to the sender that is also closer to the destination than the sending node. If all nodes employ NFP, the probability of packet collision is reduced significantly. Therefore, the average progress of the packet, which is expressed by $p^*T(a, b)$, may be higher for NFP than for MFR, (where p is the possibility of a successful transmission and $T(a, b)$ is the progress when the packet is forwarded from a to b.)

Random progress method (RPM) adopts another strategy to forward a packet [NK]. In RPM, packets destined for D are routed with equal probability toward one intermediate 1-hop neighbor node that has positive progress. A positive progress has a forward direction and a negative progress has a backward direction. The rationale for this approach is that the probability of collision grows with the distance between nodes if all nodes are sending packets frequently.

The *greedy routing* proposed by Finn is based on a geographic distance [F]. The sending node S selects a next hop that is closest to the destination, D, among all of its 1-hop neighbors. The *greedy routing* scheme only chooses a node that should be closer to D than the current sending node. Otherwise, there is a lack of advancement and the scheme fails. This algorithm does not guarantee delivery. And it has no loop because it always forwards packets a step closer to the destination.

Compass routing (DIR) uses a direction to forward packets [KSU]. In DIR, the source or an intermediate node uses the location information of the destination, D, to calculate its direction. The packet is forwarded to the 1-hop neighbor whose direction is closest to the destination D. The direction is measured between the two vectors: from sending node to the next hop node and from sending node to the destination node, D. The DIR routing scheme is not loop–free.

Geographic Distance Routing (GEDIR) is a variant of the *greedy routing* scheme [SL]. GEDIR considers all of its neighboring nodes for routing. In this variant, a packet is dropped if the best choice for a current node is to return the message to the node the packet came from. The stoppage criterion of GEDIR indicates a lack of advancement. *GEDIR routing* scheme is loop-free.

Position-based routing schemes have high delivery rates for dense networks and low delivery rates for sparse networks. If packets are successfully delivered, the performance of these routing schemes can match that of the *shortest path algorithm*.

2.5 Ad hoc network routing taxonomy

In [MC], a number of qualitative and quantitative independent properties for judging the performance of the routing protocols of ad hoc networks were listed. These important properties include: distributed operation, loop-freedom, scalability, energy consumption and guaranteed delivery. An effective routing protocol should satisfy some or all of the properties.

a) Scalability

Routing schemes should perform well for wireless networks with an arbitrary number of nodes. Scalability is an important performance evaluation for ad hoc routing schemes. Scaleable routing schemes will perform well in large networks. It has been experimentally confirmed that routing protocols that do not use geographic locations in the routing decision [JPS, L], such as AODV, DSDV or DSR are not scalable. Scalability is required for large networks. It has been experimentally confirmed that non-distributed operation routing protocols are not scalable.

b) Loop-freedom:

Loop-freedom can avoid having a small fraction of packets spinning around in the network. Loop-freedom should be an inherent characteristic for routing schemes. Otherwise, some mechanisms are needed so that packets can exit. Traditional strategies are timeout or memorizing past traffic, but they are not suitable for ad hoc networks because of energy limitations. Dijkstra's *shortest path algorithm* is an example of loop-freedom. *Greedy routing* algorithm is also an example of loop-freedom. MFR and GEDIR can provide loop-freedom. However, the DIR routing protocol does not provide loop-freedom.

c) Distributed operation

A distributed operation doesn't need global information. It only uses local information. So topology-based routing protocols are not distributed operations. *Shortest path algorithm* also isn't an example of a distributed operation. Localized routing schemes, which don't need global information, are distributed algorithms. In a localized routing algorithm, each node forwards packets only depending on its own location, its neighbors' locations and the destination's

location. A distributed operation has good scalability, as any change in a part of the network will not affect the whole network. In ad hoc networks, each node may change its status between active or sleeping and it may move arbitrarily. These will result in the frequent changes of the network topology. A non-distributed operation cannot handle these situations because of the bandwidth limitations and power constraints of ad hoc networks. *Greedy routing*, MFR, GEDIR and *FACE routing* protocols are distributed operations while *shortest path*, *shortest power path*, *shortest cost path* and all other topology-based routing protocols are non-distributed operation [S].

d) Forwarding Strategy

When a node wants to send packets, it is necessary for the node to determine the next hop for the packets. The source node can use a location service for this task. There are three main forwarding strategies for position-based routing: greedy forwarding strategy, restricted directional forwarding strategy and flooding forwarding strategy.

In greedy forwarding strategy, a node forwards a given packet to one 1-hop neighbor according to some rules. The selection of the neighbor in the greedy forwarding strategy depends on the optimization criteria of the algorithm. The greedy forwarding strategy includes MFR, GEDIR and *GFG routing*. *GFG routing* uses *greedy routing* strategy as much as possible.

A greedy forwarding strategy may fail to find a path between a source and a destination, even though one such route does exist. An example of this problem is shown in Figure 2. When a packet arrives at node *S*, it finds that *S* is the closest node to the destination, *D*, than any other neighbor nodes. The greedy forwarding strategy therefore has reached a local maximum point from where it cannot recover. The *GFG routing* algorithm, which runs *greedy routing* algorithms as much as possible, uses a recovery mechanism (FACE mode) when a packet arrives at such a local maximum point. It returns to greedy scheme again when packets reach a node that is closer to the destination than the node where the packets enter the recovery mode. The greedy forwarding strategy is both efficient (with a communication complexity of $O(\sqrt{n})$, here n is the number of total nodes) and very well suited for use in ad hoc networks with a highly dynamic topology.

A restricted directional forwarding strategy forwards a given packet to more 1-hop neighbors. A node *A* may transmit a message to several neighbors whose directions (looking from *A*) are closer to the direction of the destination, *D* than node *A*. The restricted directional forwarding strategy requires that nodes memorize past traffic. This can avoid forwarding the same message more than once and control the flooding effect.

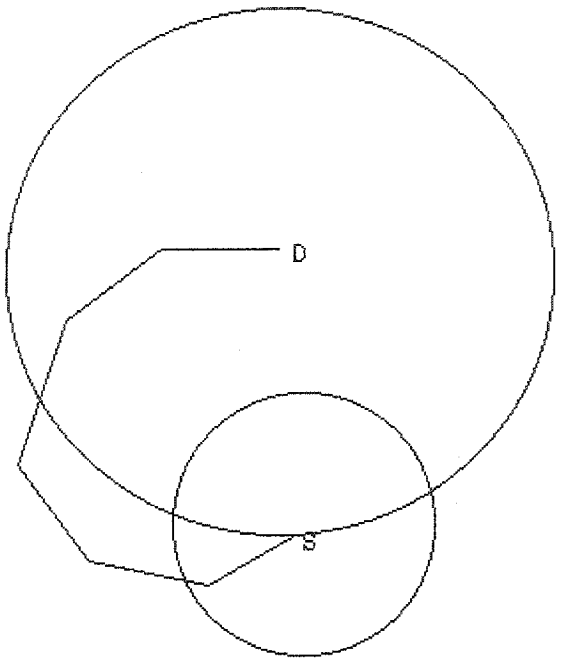


Figure 7. *Greedy routing* scheme may fail

A restricted directional forwarding strategy has a communication complexity of $O(\sqrt{n})$. It does not scale to large networks with a high volume of data transmissions. This strategy is very useful in preventing the failure of individual nodes and in preventing position inaccuracy; it is also very simple to implement. This routing scheme is suitable for applications that require high reliability and fast message delivery for infrequent data transmission.

e) Hierarchical routing scheme

Hierarchical routing schemes can reduce the complexity of networks and it allows networks to scale to a large number of nodes. *Zone Based Routing* and *Dominating Set Based Routing* are two kinds of hierarchical routing schemes.

Hierarchical routing schemes combines node locations and the hierarchical network structure. In *Zone Based Routing*, a network is divided into zones. S*hortest path algorithm* is used for inner-zone routing and location-based routing scheme is used for inter-zone routing. Each node keep a

record of the location of each zone (by treating it as a destination node positioned in the center of that zone). Routing begins by sending the message to the destination if it is in the same zone as the sender. Otherwise, the sender initiates the search for the destination by sending route requests, one to each other zone. The zone that contains the destination will reply to the sender node with the exact coordinates of the destination. The sender then learns the path to the destination and sends the full messages toward the destination using the inter-zone routing.

Dominating set based routing is also a hierarchical routing. A *dominating set* is a set of nodes. Any node in the network is either in the *dominating set* or a 1-hop neighbor of the nodes in the *dominating set*. Different literature may have different names for the *dominating set*. In this thesis, the nodes in a *dominating set* are called *internal node*s. In *dominating set based routing*, if a source is not an *internal node*, it forwards the message to the closest *internal node*. Then, the network uses location-based schemes to forward the message toward the destination by only considering *internal node*s. The message is delivered when the destination or a node that is 1-hop away from the destination is reached.

f) Guaranteed message delivery

The delivery rate is defined as the ratio of numbers of messages received by the destinations and sent by the senders. The goal of good routing schemes is to achieve a delivery ratio that is as high as possible: Guaranteed delivery is the best case. Topology-based routing schemes can guarantee delivery, but not all position-based routing schemes can do so. GEDIR and MFR routing schemes achieve very high delivery rates for dense networks, but low delivery rates for sparse ones. The *FACE routing* scheme can guarantee delivery, but it takes a relative longer path. The *GFG routing* scheme combines *greedy routing* and *FACE routing*. It uses *greedy routing* as much as possible. The *GFG routing* scheme can match the *shortest path algorithm* on dense networks.

g) Energy consumption

One of the significant differences between wireless networks and fixed infrastructures is the concern about energy consumption. Due to limited battery power, the communication overheads must be minimized in order to maximize routing tasks. Pure proactive methods that maintain routing tables with up-to–date routing information or global network information at each node are not satisfactory solutions, especially when node mobility is high with respect to data traffic. Shortest-path-based solutions are too sensitive to any change in network topology and have a

chain effect within the whole network. So proactive routing schemes are not acceptable in ad hoc networks. Location-based routing schemes may be a better choice for ad hoc networks. They perform a reactive routing scheme and only use localized information. These strategies avoid the chain effect in proactive routing schemes and don't need frequent updates. So the localized solution is more energy efficient and more suitable to ad hoc networks.

2.6 *FACE routing* scheme on planar graph

2.6.1 *FACE routing* with after-crossing scheme

FACE routing scheme is a location-based routing algorithm that can guarantee delivery. The *FACE routing* scheme works on a planar graph. A planar graph doesn't contain any two edges that intersect with each other at a common point. In a planar graph, the whole plane is divided into many blocks. Each block is called a FACE. One FACE is outside the graph; the others are inside the graph. Figure 8 shows a planar graph. The *FACE routing* algorithm works along the perimeters of FACEs using the *right (left) hand rule*.

Right (left) hand rule: If we keep our right (left) hand on a wall in a closed room and walk forward, we will visit every wall of the room. Equivalently, if we keep our right (left) hand on a wall outside a house, we will visit every outside wall of the house.

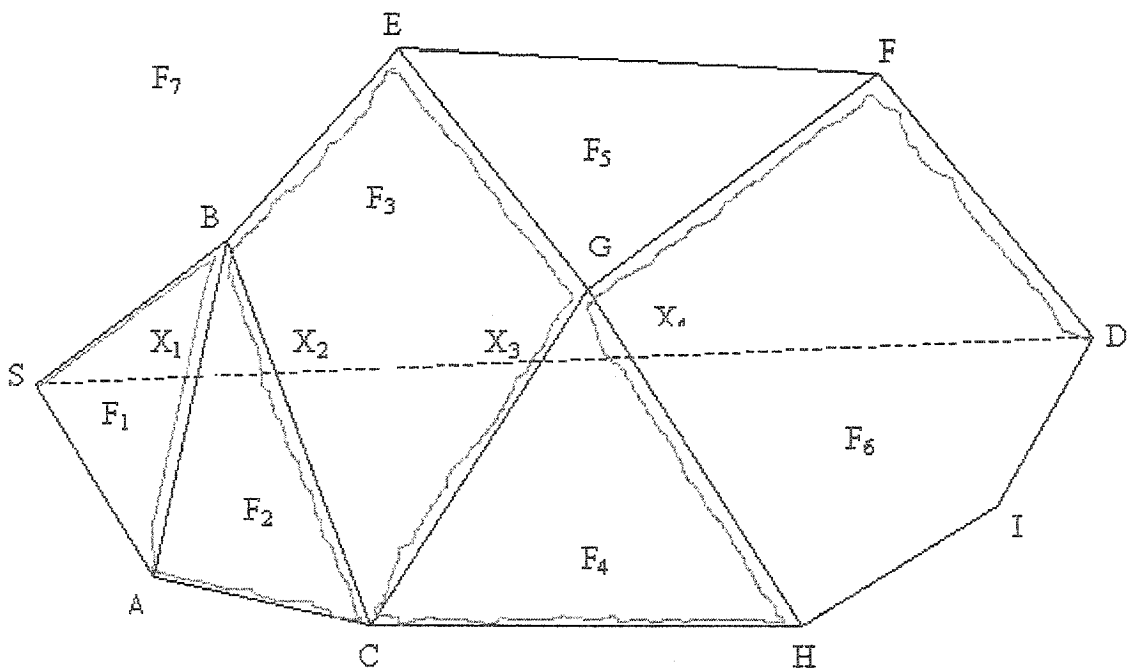


Figure 8. An example of *FACE routing* with after crossing scheme

We can use this principle on a planar graph where FACE can be seen as a room. Two FACEs share one edge in the planar graph. For any two nodes S and D on a planar graph, if we draw a line between S and D, line SD will traverse a bunch of adjacent FACEs in between node S and node D. In Figure 8, line SD traverses FACE F_1, F_2, F_3, F_4, and F_6. Line SD intersects these edges at a bunch of points: S, X_1, X_2, X_3, X_4 and D. S and D are two special points; other points are in between S and D. Any adjacent pair of intersections will share one FACE: S and X_1 are in F_1, X_1 and X_2 are in F_2 ... X_4 and D are in F_6.

The *FACE routing* scheme begins from node S. It can find the first intersection point X_1 using the *right (left) hand rule*. Then from X_1, the scheme can find next adjacent intersection, X_2. We repeat the procedure and finally we get to the destination node, D.

The *FACE routing* scheme can be summarized as follow:

> For any pair of source node, S, and destination node, D, on a planar, P, the segment, SD traverses the planar graph through a number of FACEs, F_1, F_2, ... F_N , and has a number of intersections, X_1, X_2, ... X_N. The packet travels from S in the first FACE, F_1, according to the *right hand rule* (or the *left hand rule*, as decided by some regulations) until the packet first passes the line SD at point X_1 where the length $|X_1D|$ is shorter than that of former intersection $|YD|$ (at first $|YD| = |SD|$). Let $Y=X_1$, then the packet travels in the adjacent FACE F_2 that is closer to the destination than the former one. The travel direction is also changed from counterclockwise to clockwise or vice versa. The path continues until the packet encounters new intersections X_i (i=1,2... N) with the line SD which gets closer and closer to D than previous ones.

According to above principle, the path from S to D in Figure 8 will be:

In F_1: $S - B - A$ (Applying *left hand rule* first)
In F_2: $A - C - B$
In F_3: $B - E - G - C$
In F_4: $C - H - G$
In F_6: $G - F - D$

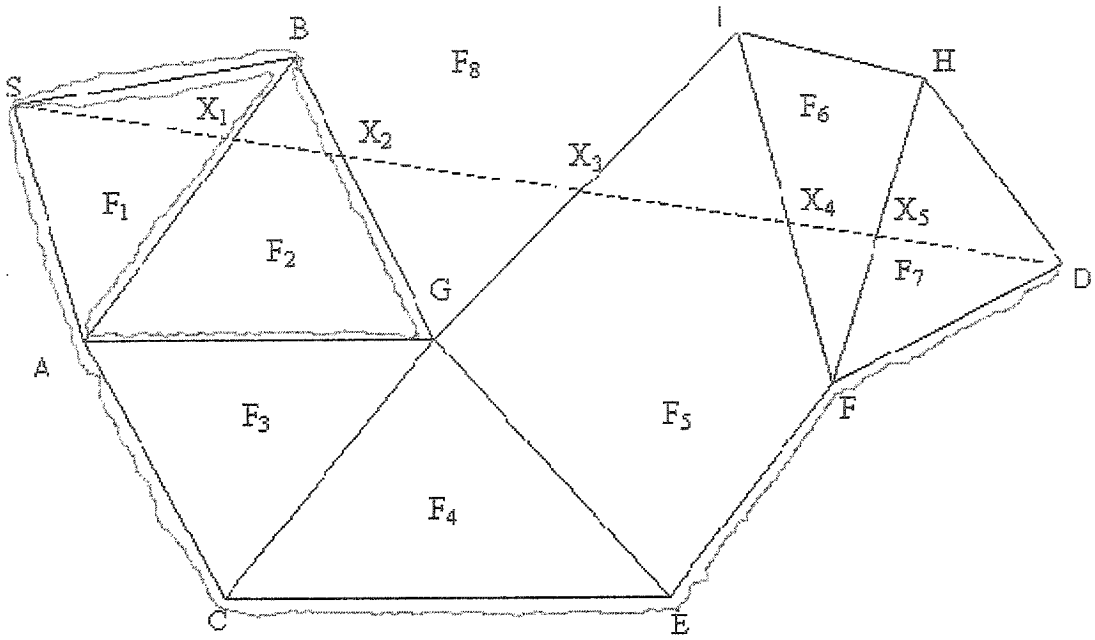


Figure 9 *FACE routing* scheme on outer perimeter

In a convex graph, line *SD* only intersects FACEs which are inside the planar graph, *P*, as in Figure 8. In a concave graph, line *SD* may intersect a FACE that is outside the planar graph *P*, as with line *SD* in Figure 9.

In Figure 9, line *SD* intersects FACE F_8, which is outside the planar *P* after line *SD* traverses FACE F_1 and F_2. When packets travel in counterclockwise direction in F_2 and intersect the line *SD* at X_2 where the intersection is closer to *D* than former intersection X_1, packets will change FACE from F_2 to F_8 and their traveling direction from counterclockwise to clockwise. The trick here is when you travel inside a FACE using the *right hand rule*, you will go in a counterclockwise direction; but when you travel outside a planar graph using the *right hand rule*, you will go in a clockwise direction. So the path from *S* to *D* on Figure 9 will be:

In F_1: $S - B - A$ (Applying *left hand rule* first)
In F_2: $A - G - B$
In F_8: $B - S - A - C - E - F - D$

Another aspect on *FACE routing* is to decide which way to go when line *SD* passes a node in a planar graph, like line *SD* in Figure 10.

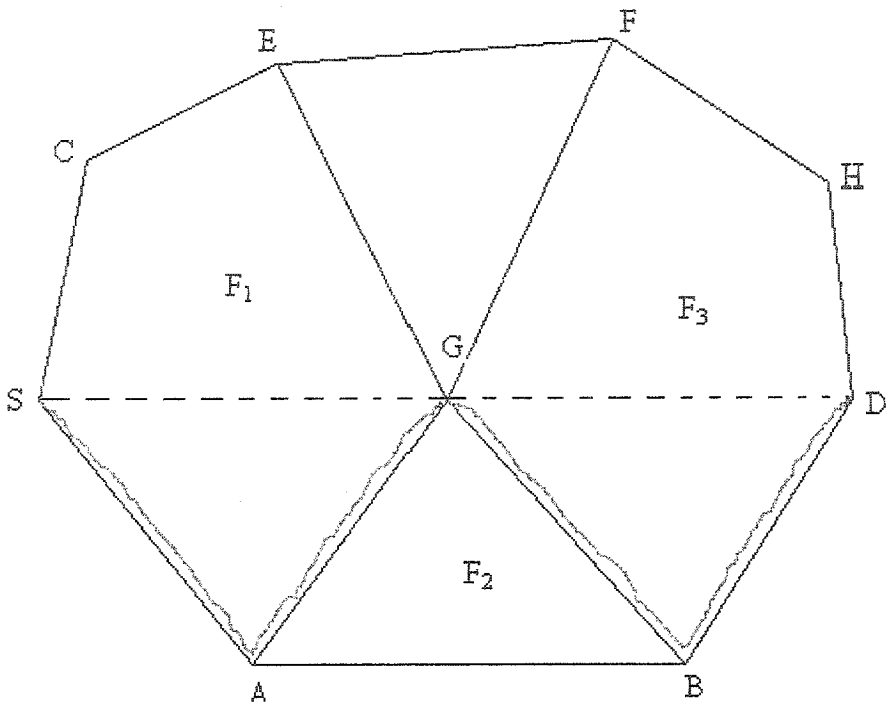


Figure 10 An example of *FACE routing* when line *SD* passes a node in a planar graph

In Figure 10, node *G* is on the line *SD*. *G* is a common point among F_1, F_2, F_3 and F_4. In this case, the packet can't decide which FACE to go next by the application of the former rules. Under this circumstance, the packet can just begin a new *FACE routing* scheme beginning from the node *G*, treating *G* as a source, and then beginning to find the next step. So the path from *S* to *D* in Figure 10 is: *S-A-G-B-D*.

An important issue to be settled here is whether to begin the first step by the *right hand rule* or by the *left hand rule*. The first step should be chosen within the FACE that is intersected by line *SD*. Assuming node *B* is the first step, deciding which node is node *B* is dependent on the angle $\angle BSD$. Here angle $\angle BSD < \pi$ and angle $\angle BSD$ can be measured in a clockwise direction or a counterclockwise direction depending on which value is smaller. The node *B* should have the smallest value of angle $\angle BSD$. If the angle $\angle BSD$ is measured in a clockwise direction, the routing algorithm uses the *left hand rule* first; otherwise, the routing algorithm uses the *right hand rule* first.

The most important thing in *FACE routing* is to know when and where to change from one traveling FACE to another. The new FACE should be closer to the destination, *D*, than the former one. The decision should be made in a localized manner. This change of traveling

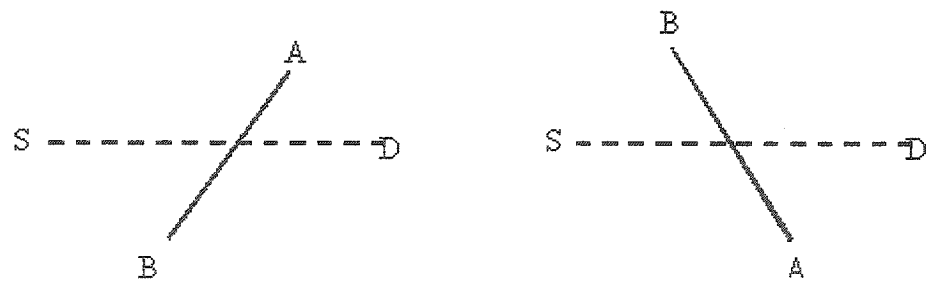
direction is important to prevent a loop on the path finding. The transition point is on the edge where a packet finds a new intersection with line *SD*. The new intersection should be closer to the destination, *D*, than the former one (*S* is assumed to be the first intersection with line *SD*). If the new intersection is further to the destination than the former one or the intersection is outside the segment *SD*, packets don't change the current FACE and continue forwarding according to the *right (left) hand rule*.

The travel direction can be determined by the following way:

> Let ***BA*** be the vector from *B* to *A* and ***n*** be a normal vector to ***BA***. If ***BA***=(*x*, *y*), then ***n*** will be (-*y*, *x*). Edge |***BA***| intersects line *SD* at a point, *X*, that is closer to the destination than the former intersection. We use the dot product ***n*•*BD*** to determine the travel direction. If dot product ***n*•*BD*** is negative, then the next travel direction will be counterclockwise. If dot product ***n*•*BD*** is positive, then the next travel direction will be clockwise.

So whenever the intersection happens and conditions for changing direction are valid, we check the dot product and determine the next direction.

The dot product ***n*•*BD*** can also be explained if node *D* is on the right side of the vector ***BA***. If *D* is on the right side of ***BA*** then the dot product ***n*•*BD*** is negative; otherwise, the dot product ***n*•*BD*** is positive.



a. D is on the right of vector BA b. D is on the left of vector BA

Figure 11 A scenario of deciding variable *sign*

In the program, a variable *sign* is used to represent the direction. Let *sign*=1 represent counterclockwise direction and *sign*=-1 represent clockwise direction. When packets travel in a planar graph, they carry the coordinates of the source, *S*, the coordinates of the destination, *D*,

and the variable *sign*. Packets can decide their next steps by using these parameters. Packets can judge whether their next hop intersects the line *SD*. Packets can calculate their new *sign* if the next hop intersects the line *SD*.

2.6.2 *FACE routing* with the before-crossing scheme

The *FACE routing* scheme we discussed before takes the routing strategy that the path changes the FACE after the packet crosses the line *SD*. This means the traveling path will have a snake-like shape. A snake-like path results in more hops. The traveling packet may continue in the next FACE without crossing the line *SD*, as shown in Figure 12. If we use the old version of *FACE routing* to find a path from *S* to *D*, the found path is: *S-F-B-C-G-H-E-D* (pink line).

There are a total of 7 hops on the path. The sending node can determine whether its next hop crosses the line *SD* by only using localized information. The traveling path can shift to the adjacent

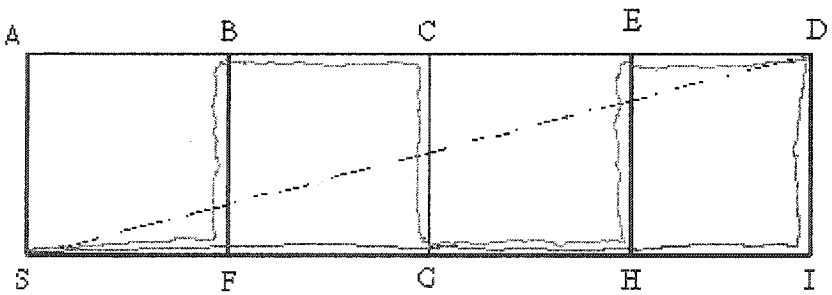


Figure 12 An comparison of two *FACE routing* schemes (Pink: first scheme; green: second scheme)

FACE before an intersection happens [BMSU]. We use the revised version of the *FACE routing* scheme to find the path from *S* to *D* on Figure 12. The found path is: *S-F-G-H-I-D* (green line).

There are only 5 hops on this path. The reduction of hops is significant, especially for longer paths. The found path by *FACE routing with before-crossing* scheme avoids the snake-like path and adopts a relatively straight one on Figure 12.

Hereafter we use *FACE routing* to refer to *FACE routing with after-crossing* scheme and new version scheme to refer to *FACE routing with before-crossing* scheme.

The procedures of *FACE routing with before-crossing* scheme are mostly the same as the old version [BMSU]. The new version scheme also works on a planar graph. If the traveling packet doesn't cross the line *SD*, the path found by the new version is the same as the path found by the old version. If a sending node predicts the next hop will cross the line *SD*, the current FACE will be changed to the adjacent new FACE. The next hop can be decided in following ways:

First, the new version scheme detects the intersection point *X*. If *X* is out of the segment *SD*, the traveling packet ignores the intersection and continues in the same FACE. Otherwise, the new version scheme compares the current intersection *X* with former intersection *Y* (at first, let *Y=S*). If $|XD| \geq |YD|$, the traveling packet keeps going forward without changing. If $|XD| < |YD|$, then the traveling path will be shifted to the adjacent new FACE.

Let *A* be the current sending node. *A* will send packets to *B* according to the *right (left) hand rule*. Now *A* predicts an intersection with line *SD* if it sends packets to *B*. At this time, *A* will not forward packets to *B*; instead, *A* will check node *C* which has the smallest angle $\angle BAC$. If there is no intersection between line *SD* and edge *AC*, *A* will forward packets to *C*. Otherwise, *A* will continue checking the next node *E* that has the smallest angle $\angle CAE$. This procedure continues until *A* finds a qualified node. *A* may send packets to its former node if no qualified node is found. But this will not lead to a loop. When *A* sends back to its former node, this node will choose another node to go to. Figure 13 shows an example of such a situation. The path from *S* to *D* is: *S-A-S-B-I-G-H-D.*

The first step of *FACE routing* with the before-crossing scheme is the same as the first version. The variable *sign* in the new FACE algorithm may have a different situation. The path from the source to the destination may not intersect the line *SD*. So the variable *sign* will not be changed.

FACE routing with before-crossing scheme may not always be better than the old version. An example is shown in Figure 14.

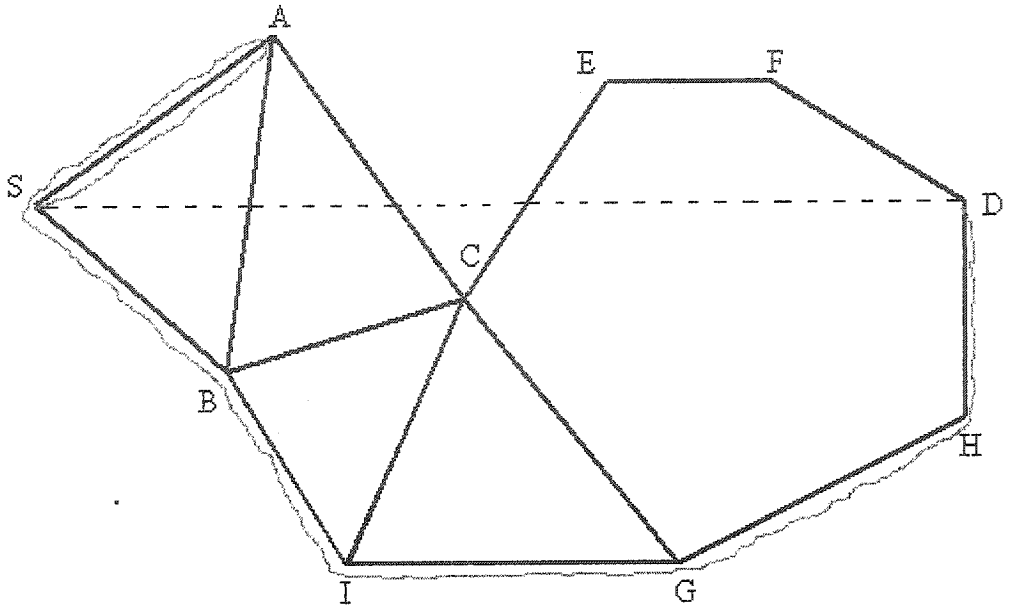


Figure 13 *FACE routing with before crossing* scheme chooses the outer perimeter

Figure 14 shows a planar graph with snake-like shape. There is only one path from the source to the destination. There is no alternate path for packets to go to. When S sends packets to A and A predicts an intersection X_1 if it sends packets to B, A will forward packets back to S and attach the intersection point X_1 with packets. S sends packets to A again. This time A will forward packets to B. Although there is an intersection, X_1, between line SD and edge AB, the condition $|XD| < |YD|$ doesn't hold, here X is the current intersection and Y is the former intersection. When B wants to forward packets to C, B predicts a new intersection, X_2. Let $X=X_2$ and $Y=X_1$. Because of $|XD| < |YD|$, B will send packets back to A. The former procedure repeats again. The basic principle here is the *right (left) hand rule*. The complete path found by *FACE routing with before crossing* scheme is: *S-A-S-A-B-S-A-B-C-D*. There are 10 hops in the path.

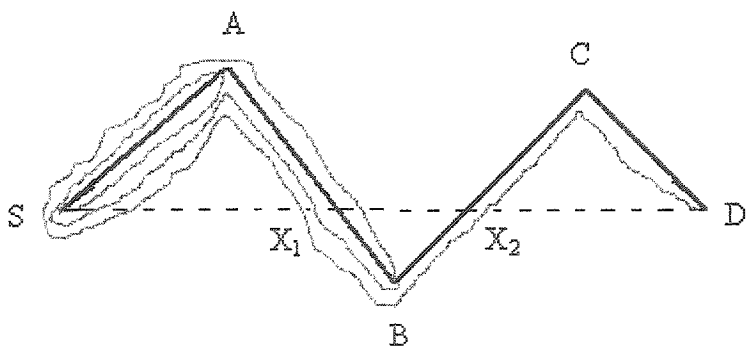


Figure 14 A snake-like graph

If we use the old version FACE scheme to find a path from *S* to *D*, the complete path is: *S-A-B-C*. There are a total of 4 hops. *FACES routing with before-crossing* scheme and the old version FACE scheme have different performances on different UDGs. If a UDG has a relatively small number of total nodes, *N*, and comparatively big average degree, *d*, the new version scheme may have a better performance than the old version scheme because there will be less snake-like paths in this kind of UDG. On the other hand, the old version scheme has a better performance than the new version scheme if the UDG has a bigger number of nodes, *N*, and smaller average degree, *d*. There are many snake-like paths in these kinds of UDGs

2.7 *GFG* routing with guaranteed delivery

The *GFG routing* scheme that combines greedy *mode* and FACE mode is proposed in [BMSU]. The *GFG routing* scheme first runs greedy mode as much as possible. But the greedy mode can't guarantee delivery. When the greedy mode encounters a local maximum point, the *GFG routing* scheme fails. At this moment, the *GFG routing* scheme changes to the FACE mode scheme. The *GFG routing* scheme runs FACE mode scheme until the packet encounters a node that is closer to the destination than the local maximum point. The *GFG routing* scheme switches back to the greedy mode scheme again. The alternation between the greedy mode and the FACE mode scheme may happen several times. The path finding procedure terminates whenever either routing scheme finds the destination. The *GFG routing* scheme has a much better performance than the pure *FACE routing* scheme. Its performance is very close to the *shortest path algorithm* scheme on dense networks. The *GFG routing* scheme can guarantee delivery.

2.8 *Dominating sets* based routing

Datta [DSW] proposes some improvements on the *FACE routing* scheme and the *GFG routing* scheme. These improvements include *dominating set*. The proposal uses a *dominating set* to construct a connected sub graph. The sub graph only contains nodes in the *dominating set*. Then *FACE routing* scheme and the *GFG routing* scheme are implemented on the sub graph. Packets first find the closest *internal node*. Then packets are forwarded within *internal nodes*. The path finding procedure ends when the packets either get to the destination or the packets get to an *internal node* that is one hop away from the destination.

Dominating set can reduce hops in path finding when routes are only selected within a *dominating set*. Several kinds of *dominating set* notions are proposed which have different contents. In this thesis, two kinds of *dominating sets* are used. They are *intermediate nodes* and *inter-gateway nodes*. Wu and Li in [WL] introduced these two kinds of internal nodes.

a) *Intermediate node*

Definition: A node, *A*, is an *intermediate node* if there are two neighbors *B* and *C* that are not direct connect neighbors themselves.

Another kind of *internal node* is *inter-gateway node*. *Inter-gateway nodes* are generated from *intermediate nodes*. *Inter-gateway nodes* further reduce the number of nodes from a UDG graph.

b) *Inter-gateway node*

Definition: Nodes u and v are *intermediate nodes*. Let $N_1[u]$ denote the 1-hop neighbors of u and include node u itself. $N_1[v]$ denotes the 1-hop neighbors of v and includes node v itself. If $N_1[v] \leq N_1[u]$ and $ID(v) < ID(u)$, then node v is excluded from the *inter-gateway node* set. Applying this rule to every node in the *intermediate node* set, the left nodes are called *inter-gateway nodes*.

In the definition, the ID of a node can be a random number in [0,1), or can be the sequential number assigned to each node during the procedure when we generate a UDG. The ID can also be other values. In this thesis, we use the sequential number of each node as the ID of the nodes.

The reasons why we use *dominating set* are because *dominating set* has the following properties: If the UDG has *dominating set* nodes, any node in the UDG is either a *dominating set* node or a direct neighbor to some *dominating set* nodes. In other words, a node is either a *dominating set* node or one hop away from the nearest *dominating set* node.

Let *G* be the graph of UDG and *G'* be the *dominating set* graph extracted from *G*. The nodes in *G'* keep the same connections as they have in *G*. Then graph *G'* is connected among *dominating set* nodes.

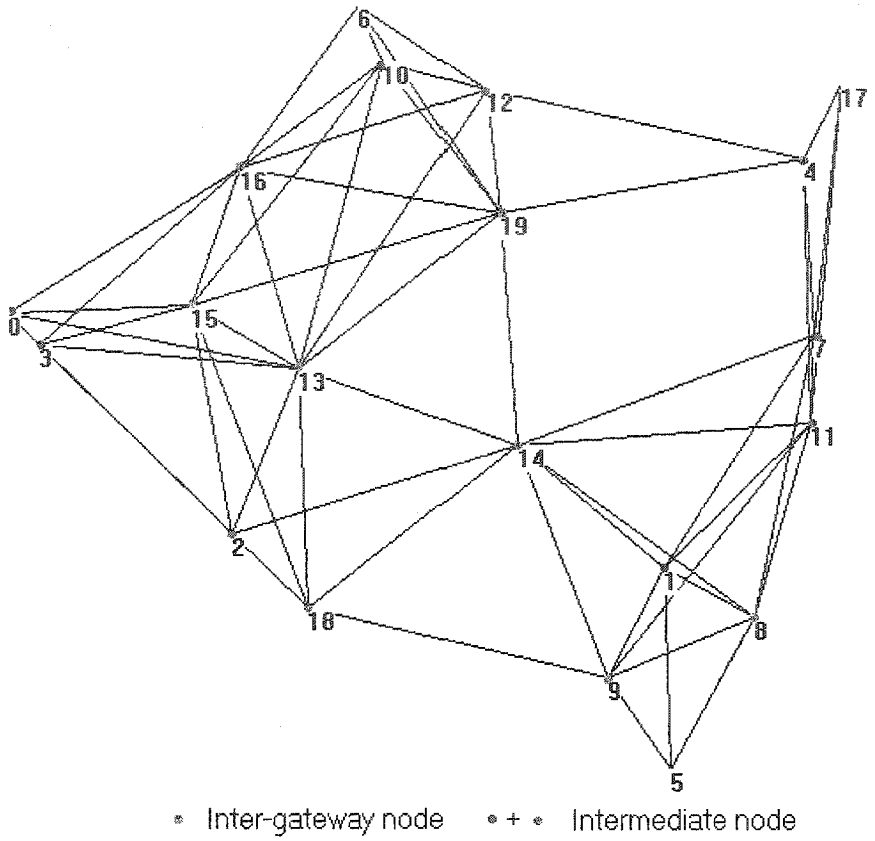


Figure 15. A scenario of *internal nodes*

FACE routing scheme incorporated with the *dominating set* will lead to more effective path finding. Because a *dominating set* sub graph is only connected among *dominating set* nodes, to implement *FACE routing* scheme and *GFG routing* scheme from any source node *S* to destination node, *D*, in UDG, we have to construct a partial UDG on *S* and *D*. A partial UDG on pair *S* and *D* is constructed by following way:

> *Input (UDG, S, D)*
> *Output (partial UDG)*
> *First, extract a dominating set sub graph G' from UDG;*
> *Second, for any pair S and D, we append the connections among nodes S, D and the dominating set if these connections exist in the UDG. We also append the connection between pair S and D themselves if the connection exists in the UDG;*
> *After the two steps, a partial UDG on pair S and D is constructed.*

The constructed partial UDG on pair *S* and *D* is connected between node *S* and node *D*. In other words, a path can be found between node *S* and node *D*.

The following is the pseude-code to implement the routing algorithm on a *dominating set*.

Route_IG_Algorithm(S, D)
Input (S, D, UDG)
Output (path between S and D)
/ find path between S and D by using dominating set */*
Construct dominating set sub graph G' from UDG;
Add S and D into the dominating set if they are not in the dominating set;
Add all edges among S, D and dominating set nodes if these edges exist in UDG;
Implement GG, DT or PDT on the new constructed graph;
Implement FACE(S, D) or GFG(S, D) on the constructed sub planar graph.

2.9 Shortcut scheme on *FACE routing*

FACE routing works in a localized manner. But routes found by *FACE routing* have more hops than topology-based routing schemes or *greedy routing* scheme. The *FACE routing* takes a snake-like shape from the source to the destination. This is why the path taken by *FACE routing* is relatively longer. According to the definition of a GG graph, we can generate an example as shown in Figure 16 that combines a UDG and a GG graph on one graph. The green edges consist of a GG graph. All edges belong to a UDG. The *FACE routing* scheme is used to find the path between node 3 and node 5. The found path is 3-2-13-14-18-9-5.

The path takes 2 hops from node 3 to node 13 and 2 hops from node 13 to node 18 although these pairs of nodes have direct connections. In a GG graph, these connections don't exist. This phenomenon also exists in DT graphs and PDT graphs. If any node in the UDG can have 2-hop neighbor information, the current sending node can decide which node to forward packets to and predict the next hop that the receiving node will take. In the above example, node 3 forwards packets to node 2 and predicts node 2 will forward packets to 13, node 13 forwards packets to 14 and predicts node 14 will forward packets to 18 if the two nodes have 2-hop neighbor information. Node 3 can send packets to node 13 directly and node 13 can send packets to node 18 directly to make shortcuts. A shortcut scheme reduces hop count.

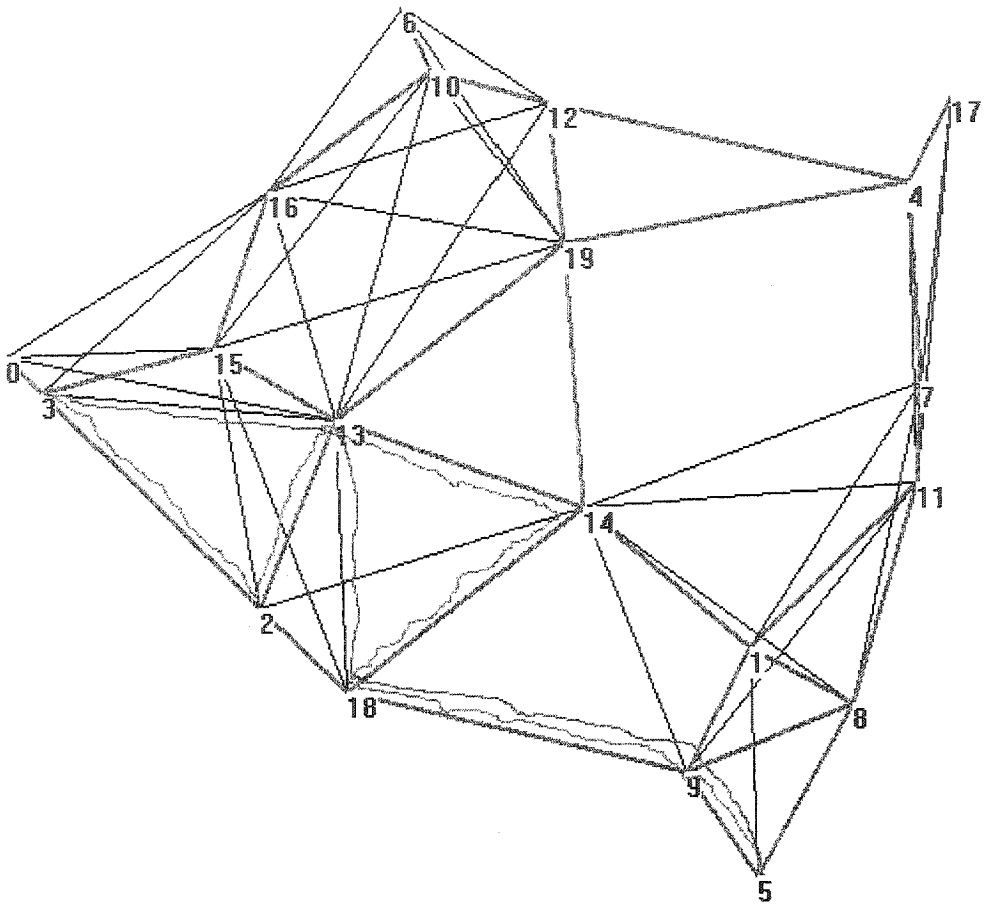


Figure 16 An illustration of shortcut scheme

(The GG and the UDG are in one graph. Green edges belong to the GG. Pink: path found by *FACE routing* scheme; red: path found by *FACE routing* scheme with shortcut)

2.10 Address configuration

Address configuration is a difficult problem in ad hoc networks. Many proposals and algorithms are proposed for address configuration in ad hoc networks. Auto-configuration is a desirable goal in implementing ad hoc networks. In traditional networks, using a DHCP server performs such a dynamic address assignment. Configuration tools provide addresses for nomadic users on wired networks. Organizations owning wide IP networks use centralized configuration servers to automatically configure their networks. This mechanism allows the dynamic configuration of transient users in the networks. Implementing DHCP, however, requires access to a DHCP server. In the case of MANET, centralized configuration architecture, such as DHCP, isn't a reliable solution because a server or nodes can be disconnected from the MANET from time to time.

Mobile IP (MIP) uses another mechanism to fulfill address configuration. MIP has been proposed in order to handle seamlessly the mobility of the nodes running the IP protocol. A mobile node *M* belongs to its *home network*, its main organization. Inside this *home network*, node *M* is referenced by a *home address*. This *home address* identifies the node permanently, even if it is not in its *home network*. When the mobile node *M* moves, it may visit a network belonging to another IP domain. This network is referred to as *foreign network*. The *foreign network* owns a particular entity named a *foreign agent*. A *foreign agent* is responsible for giving the mobile node *M* a temporary address called a *care of address*.

Nesargi and Prakash [NP] propose an approach for stand-alone MANETs called MANETConf. This approach has two-phase processes: initiation and validation. In MANETConf, when a new arriving node (the request) sends a request for an IP address to a former configured neighbor (the initiator), the initiator will broadcast an IP address request to the whole network. Any node which has the IP address in the network will answer the request. This ensures that the request will not have a duplicate IP address. If a node doesn't answer after a given number of attempts, it is considered to have left the network and its address is relinquished. MANETConf uses the lowest IP address in the network to represent an ad hoc network. It detects splitting and merger by comparing the current lowest IP address with the former lowest IP address within all its partitions.

M. Genes et al. [GR] proposes using an *Addressing Agent* as acting DHCP server. This proposal is roughly based on the election of one of the ad hoc network nodes as an addressing configuration server. The *Addressing Agent* has the lowest MAC address within the whole network and it keeps a list of all nodes. The list contains the mapping of MACs to IP addresses. The MAC address of the *Addressing Agent* is also considered the identifier of the network. The identifier is used to handle network splitting and merging. When several networks merge, the node with the smallest identifier is chosen as the new *Addressing Agent*. If the network splits, each partition will have an *Addressing Agent* that has the smallest MAC address within its part.

Zhou et al. [ZNM] proposes a *prophet address allocation* scheme for large scale MANETs. It uses a function $f(n)$ to generate IP addresses for the network. The first node *A* in the network chooses a random number as its IP address and default state value. This value is used as the seed of function $f(n)$. A new node *B* hopes to join the network, node *B* asks *A* for an IP address. Node

A uses $f(n)$ to generate a new state value and gives this value to *B* as its IP address and state value. Any new node uses the same system to get its IP address and state value. In each instance, a node updates its state value when it gives an IP address and state value to other nodes. This approach assumes that function $f(n)$ has the following two properties: (1) The interval between two occurrences of the same number in a sequence is extremely long. (2) The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some interval is extremely low. In this approach, the first node *A* has the seed and function $f(n)$. It can compute the IP address sequence locally. That means it knows in advance which address is going to be distributed. This is where the name *prophet address allocation* comes from. The first node *A* can predict a potential conflict in the IP address allocation. If there are duplicate numbers in the sequence, node *A* can choose another seed to generate other sequences to prevent IP address conflicts. The advantage of this protocol is that it avoids duplicate IP addresses and is easy to handle partitions and mergers, but this scheme needs a big address pool for IP address allocation and a good function $f(n)$ is not easy to find.

Ojeda-Guerra et al. [OMA] propose a scheme for node configuration in ad hoc networks. The main idea is to choose a node to be a DHCP superserver. At the beginning, the first and only node in the network is the first DHCP server. The first node is in charge of distributing IP addresses to other devices that wish to join the network. Any node in the network can act as a DHCP server if it runs such an application. Among them, a superserver is chosen. The superserver assigns addresses to new devices. In order to choose the superserver, each DHCP server must have information about its environment. Each DHCP server must know all of the other DHCP servers within its communication range. The superserver is the one with the lowest IP address. This approach can handle partitions and mergers, but it needs communication overheads within each DHCP sever.

Orset and Cavalli [OC] propose an automatic node configuration and address authentication protocol. It uses the concept of recursive binary trees to provide addresses to the nodes. Authentication is used for security. In this scheme, each node has the responsibility to distribute a free address range. If a node receives an address request from another node, it will divide its address range into two equal parts and give one of them to the address request. The receiving node keeps the first address for itself. It also divides the rest of the addresses into two parts and treats other requests as the same.

2.11 Data-centric storage in sensor networks

A sensornet is comprised of a large number of small devices, each device with the capabilities of computation, storage and communication. Sensornets will generate huge amount of information. While sensornets give unprecedented access to detailed information about the physical world, sending this overwhelming volume of observations directly to the access point would promptly exhaust the energy reserves of the sensornet. Making effective use of the vast amount of data gathered by large-scale sensor networks will require robust and self-organizing data storage protocols.

It is assumed that a sensornet is connected to the outside world through a small number of access points. There are three canonical methods for the sensornet data storage [KSEGYY].

External Storage: Any node will send detecting data to external storage where this data is processed. External storage has a cost of $O(\sqrt{n})$ for each observation obtained from the sensors.
Local Storage: Observation information is stored locally (at the detecting node). Inquiries are broadcast to all nodes at a cost of $O(n)$. Responses are sent back to the source of the inquiry at a cost of $O(\sqrt{n})$ each.
Data-Centric Storage (DCS): An observation is detected and the data is stored by name within the sensornet. The communication cost to store the event is $O(\sqrt{n})$. Inquiries are directed to the node that stores observations by that name.

DCS has the best performance in cases where (a) the scale of sensornet is big; (b) there is a lot of gathered data and not all data types are inquired [KSEGYY].

DCS uses the *GFG routing* scheme for the data transmission protocol. GHT is built atop *GFG routing* scheme [KSEGYY]. The critical issue in GHT is the hashing of a name, k, into geographic coordinates. A (name, value)-based associated memory is used in DCS. Both the storage and retrieval of observation data are fulfilled by using their names. There are two operations performed on the data: *Put*() and *Get*().

Put(k, x) stores x (the collected data) according to the name k.
Get(k) retrieves the data that is stored associated with name k.

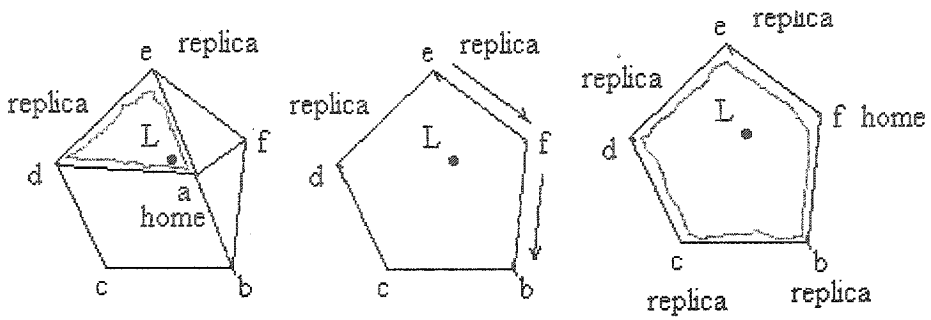
Both a *Put*() operation and a *Get*() operation on the same name *k* hash *k* to the same location. A name-value pair is stored at a node in the vicinity of the location. Choosing this node consistently for the name is crucial to build a GHT. The source of a *Put*() or *Get*() operation for a name *k* hashes the name *k* into geographic coordinates that are the destinations of the packets for that operation. The hash function is ignorant of the location of individual nodes in the sensornet. The operation of a hash function evenly distributes the different names throughout the geographic region where the sensornet is deployed. It is quite possible that there isn't any node at the precise coordinates. The hash function requires a *home node* for a GHT. A *home node* is defined as the node geographically nearest to the coordinates of the name. A DCS packet is not addressed to a specific node but to a specific location. This location is treated by *GFG routing* scheme as a disconnected destination. The *home node* serves as the destination point for *Put*() and *Get*() operations of the same name. So *GFG routing* scheme will route such packets to the appropriate *home node*.

GHT uses *GFG routing*'s FACE mode to find the *home node*. When *GFG routing* encounters a local maximum point, it switches to *FACE routing*. *GFG routing* may change back to *greedy routing* again. When *GFG routing* encounters its last local maximum point, it will tour the perimeter that encloses the destination point. The last local maximum point is the *home node*. The perimeter that embraces the destination point is the *home perimeter*. Under GHT, the *home node* knows to consume the packet if the packet returns after its tour of the *home perimeter*.

The perimeter refresh protocol (PRP) [KSEGYY] is used by DCS to provide both persistency and consistency when the topology of the sensornet changes. This protocol replicates stored date for the name, *k*, at nodes around the location to which *k* hashes, and ensures that one node is chosen consistently as the *home node* for that *k*. So all storage requests and queries for *k* can be routed to that *home node*. This protocol has some local communication overheads, especially on networks where nodes are densely deployed. By hashing names, GHT evenly spreads storage and communication loads among different names across the sensornet. The network has a fair traffic load distributing throughout the whole network.

PRP is used to store a copy of a name-value pair at each node on the *home perimeter*. All nodes on the *home perimeter* are called replica nodes except the *home node*. PRP uses a timer mechanism to generate periodical refresh packets. A *home node* sends a refresh packet addressed to the hashed location of the name. The packet will use *FACE routing* to tour the *home*

perimeter. If the packet encounters another node that is closer to the destination than the current *home node*, then this node will become the *home node*. The former *home node* may become a replica node if it is still on the *home perimeter*.



(1) Home node a (2) Node a fails (3) New home node f

Figure 17 A procedure to create a new *home node* (Red circle is the *home perimeter*)

If the *home node* fails, any replica node can initiate a refresh packet touring the *home perimeter* using *GFG routing* scheme. The refresh packet will help to find a new node that is the closest to the destination. This node will transit to the *home node*. By this way, GHT maintains the consistency of the stored data on the *home perimeter*. Figure 17 shows a procedure to find a new *home node*. In figure 17, *L* is the hashed location. Node *a* is the nearest node to *L*. So node *a* is the *home node* for *L*. When node *a* fails, node *e* initiates a PRP procedure. After this procedure, node *f* becomes the new *home node* for *L*. Node *b, c. d* and *e* are replicas.

Chapter 3 PDT based routing with guaranteed delivery

In this chapter, we present the performance evaluation of *FACE routing* and *GFG routing* on different graphs. We first generate random UDGs, then *FACE routing* and *GFG routing* with different improvement schemes are implemented on each UDG. The generated random UDGs have different degrees. GG, PDT and DT graphs are used in the routing algorithms for comparison. Existing algorithms to generate random UDGs take long time to generate sparse networks. In this chapter we propose a new algorithm to generate a random UDG which solve the problem. We use Microsoft Visual C++ to construct the simulation environment. Existing software for network simulation (at the time of beginning this project), like *NS-2*, lack scalability on simulation and they don't meet our specific demands, while others, like *Glomosym*, need long learning curve. We therefore developed new codes to simulate these routing protocols.

3.1 New method to generate UDG

We propose a new method for generating random unit disk graphs. The method is good for generating UDG with a small average degree. It is summarized as follows. (Hereafter we use degree to refer to average degree of a UDG.)

Algorithm to generate a random UDG:

Input (N, d)

Output (UDG)

/* the procedure to generate a random UDG graph*/

1. An approximate radius r is obtained from the formula $d=(N-1)*r*r/\pi*(a*a)$. The approximation is obtained by finding expected number of nodes inside circle with transmission radius (the formula multiplies number of other nodes with probability of being generated inside mentioned circle). The first node is randomly generated (that is, its *x-coordinate* and y-coordinates are chosen at random) in a given square with edge length a. Each of following N-1 nodes is generated repeatedly, at random, and tested whether it is within distance r to at least one of previously generated and accepted nodes, until the test is satisfied. Otherwise (when it is at distance $>r$ to all previously accepted nodes), the node is rejected and another node generated and tested instead.

2. After selecting N nodes with this procedure, a connected random unit graph is generated. However, its average degree q is not necessarily the desired one, d. We now find the exact average degree d of generated graph, by counting edges and compare it with desired value d. If $q<d$, more edges need to be added, and graph remains connected. We sort all $N*(N-1)/2$ possible edges and desired radius R is the $N*d/2$-th edge in sorted list. Unit disk graph is then decided using this value of R (that is, two nodes are neighbors if and only if the distance between them is at most R). If $q=d$ then the graph remains unchanged. If $q>d$, this obtained random unit disk graph is too dense, and some edges need to be deleted by reducing transmission radius. We sort all $N*(N-1)/2$ possible edges in increasing order and find the $N*d/2$-th edge in the sorted list. We use this edge as the transmission radius R, and define the corresponding graph, which may not be connected. Dijkstra's *shortest path algorithm* is used to check the connectivity of this graph. If the graph is not connected, it is ignored, and the procedure is repeated. If it is connected, which should happen with high probability with this procedure, the graph is accepted.

The new method generates a UDG faster than the first method when the required degree is small. We can also adjust the radius r for different degrees. The drawback of the new method is that the size of the UDG is usually smaller than that of the UDG generated from the first method. Thus, the UDG from the new approach usually likes a small cluster.

3.2 PDT based Routing scheme

In the experiments, we implemented the *FACE routing* scheme and the *GFG routing* scheme on GG, DT and PDT graphs. Although a DT graph can't be constructed in a localized manner, we also implemented *FACE routing* scheme and *GFG routing* scheme on a DT graph for comparison. To do comparisons, we first generated a UDG graph, then ran *FACE routing* scheme and *GFG routing* scheme on GG, DT, and PDT graphs separately. GG, DT and PDT graphs were generated from the same UDG. So for one case, nodes on GG, DT and PDT graphs have the same geographic distribution, but the edges in these graphs are different. The UDG graphs were generated randomly. Each case has 30 samples. *FACE routing* scheme and *GFG routing* scheme were implemented on these samples separately. We used these two routing algorithms to find paths between any two nodes on GG, DT or PDT graphs. The efficiency of the

routing algorithm is measured by the average hop count. We use Dijkstra's *shortest path algorithm* as the benchmark for comparison. In the simulation, the sending node was assumed to know its own location, the destination's location and all of its 1-hop neighbors' location information. For one planar graph, we found the paths between any pair of source and destination. The average hop of all paths of the UDG was given as the average hop count of the network. For example, if the UDG has 60 nodes, there are 60x59=3540 pairs of sources and destinations. We find all 3540 paths using *FACE routing* scheme and *GFG routing scheme*. The average hop of 3540 paths is the average hop count of this network. We use statistics to find the average hop of samples and 95% confidential interval (CI). The average of 30 samples is present in the tables for the corresponding cases. Figure 18 and Figure 19 show the results of the experiments. More precise data can be seen from Table 1.

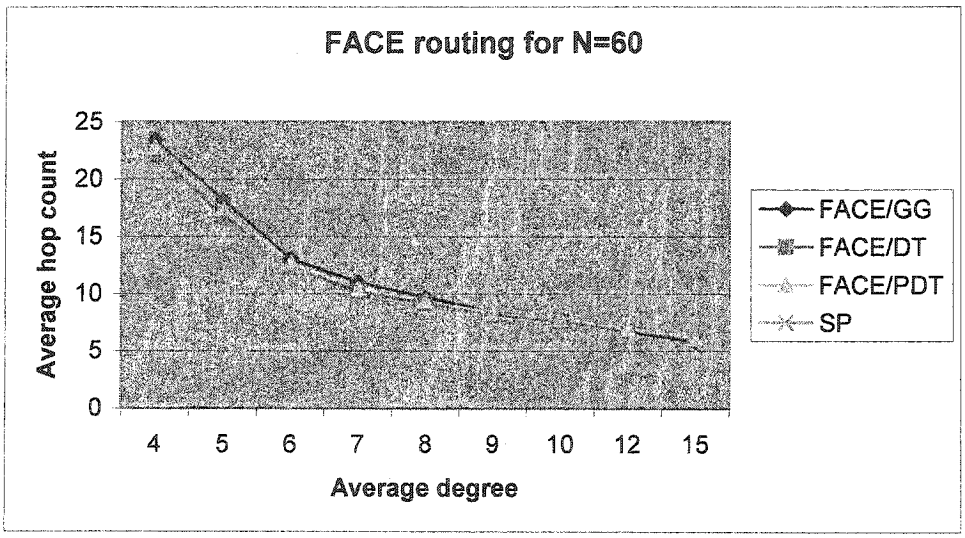


Figure 18 Comparison of *FACE routing* on different planar graphs

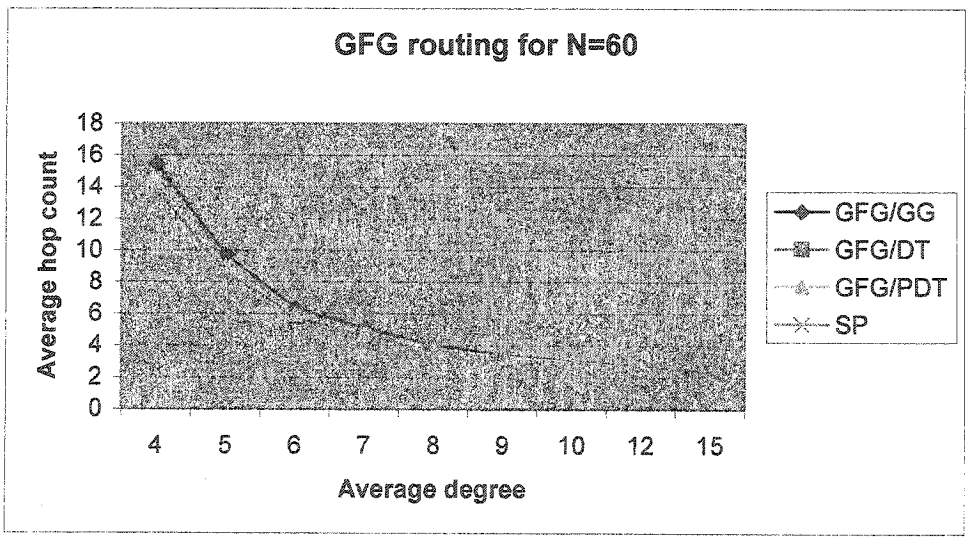


Figure 19 Comparison *of GFG routing* with different planar graphs

By looking at Figure 18 and Figure 19, we learn that the *FACE routing* scheme has better a performance on a PDT graph than on a GG graph when N=60 and degree < 10. *FACE routing* scheme on a DT graph has the best performance when degree < 10, but it has the best performance on a GG graph when degree > 10. We have mentioned that a GG graph is a sub graph of a PDG graph and a PDT graph is a sub graph of a DT graph. So a DT graph is the densest graph. For N=60, when degree is less than 10, the constructed planar graph, GG, is more likely to be concave. When we constructed a PDT and a DT, the added edges in the DT and the PDT are more likely than the GG to be on the outer perimeter of the graphs. So the new added FACEs are also more likely to be on the outer perimeter. If we draw a line between any pair of nodes in the PDT and the DT, this line may not traverse these new added FACEs. So the added FACEs in the PDT and the DT may not result in more hops between the two nodes. On the other hand, the added edges can act as shortcuts for some paths. Thus, the total hops for some pairs of nodes reduce. In contrast, for N=60 and degree > 10, the generated UDGs are more likely to be convex. When we construct a PDT and a DT graph, the added edges are more likely to be at the intermediate of the graphs than the GG. So those added FACEs in the PDT and the DT than the GG are also at the intermediate of the graphs. This results in more FACEs to traverse for some pairs of nodes. That means more hops for the paths between some pairs of nodes. Under these circumstances, a GG graph has the lowest average hop count.

GFG routing scheme has a different situation than *FACE routing* scheme. *GFG routing* scheme always has the best performance on a DT than on other graphs. It has better performance on a PDT than on a GG. Because *GFG routing* runs a greedy scheme first, more edges in a graph mean more options to choose and more shortcuts. That is why the *GFG routing* scheme on a DT has smallest hop count.

3.3 PDT based Routing with *inter-gateway* scheme

In the second step, we implemented the concept of *inter-gateway node* into our routing algorithms. For any generated UDG, we first looked for *intermediate nodes* in the UDG, then we found the *inter-gateway nodes* in the UDG. These *inter-gateway nodes* consist of a small part of the total nodes in the UDG. To find a path between any pair of nodes, a partial UDG graph was constructed. The partial UDG graph consists of the *inter-gateway nodes*, the source node and the destination node. Then, we constructed a GG, a DT or a PDT graphs on this partial UDG graph.

This procedure needs to be repeated for any pair of source and destination because different pair of sources and destinations may have different partial UDG graphs. *FACE routing* scheme and *GFG routing* scheme are fulfilled on these planar graphs. Figure 20 shows the results of average hop. More precise data can be seen from Table 3.

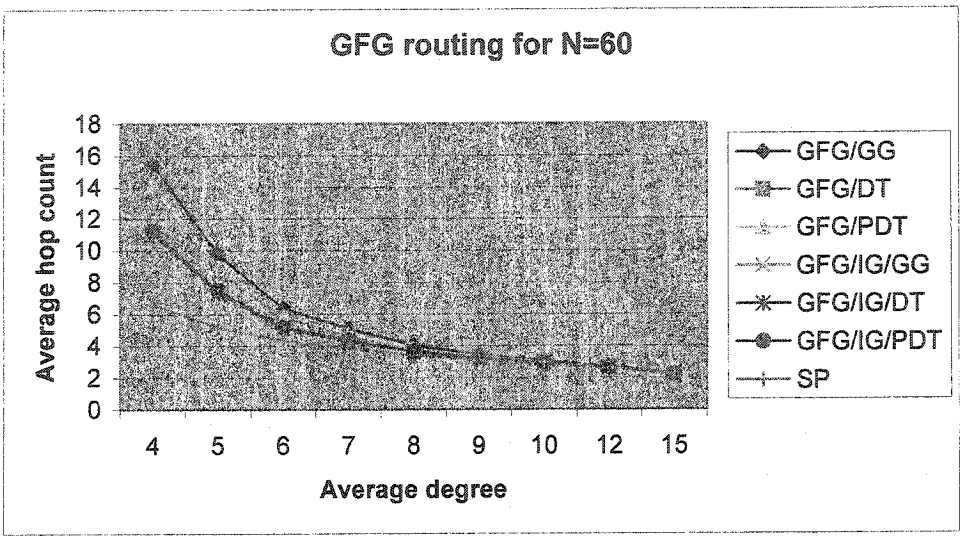


Figure 20 Comparison of *GFG routing* with *inter-gateway node* scheme

By looking at Figure 20, it can be seen that the *inter-gateway node* scheme significantly reduces the average hop count, especially on a low degree UDG (For N=60, degree < 10). The *inter-gateway node* scheme reduces the average hop count when using the *GFG routing* scheme. On degree=4, when planarized by a PDT graph, the average hop count reduces by 20.4%; the average hop count reduces by 4.7% on degree=8; and the average hop count only reduces by 1.0% on degree=15. It can be concluded that the *inter-gateway node* scheme has better effects on a small degree UDG. We compare the average hop on a GG, a DT and a PDT graph. They are pretty close. Considering the statistical deviation, there is not a big difference between the average hop count of GG, DT and PDT graphs when using *inter-gateway node* scheme.

3.4 PDT based Routing with shortcut scheme

In the third step, we implemented the shortcut notion into the routing scheme. We used the shortcut on the routing scheme with an *inter-gateway node* scheme or without *inter-gateway node* scheme. Figure 21 and Figure 22 show the results of average hop count. More precise data can be seen from Table 5.

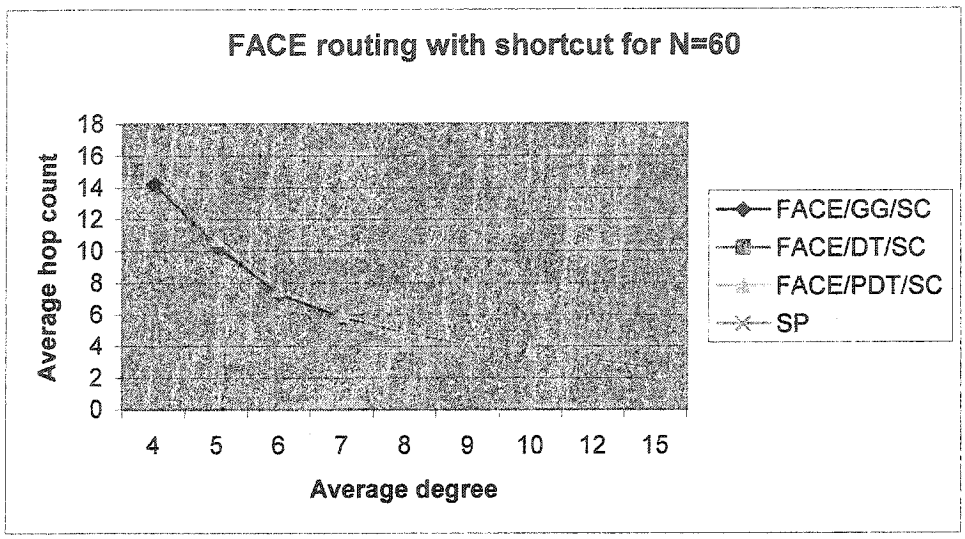


Figure 21 *FACE routing* with shortcut scheme on different planar graphs

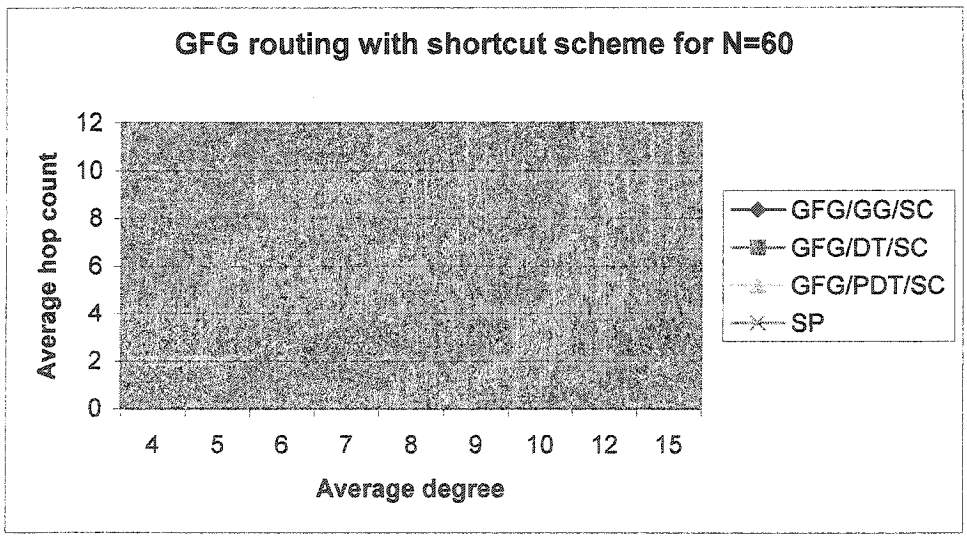


Figure 22 *GFG routing* with shortcut scheme on different planar graphs

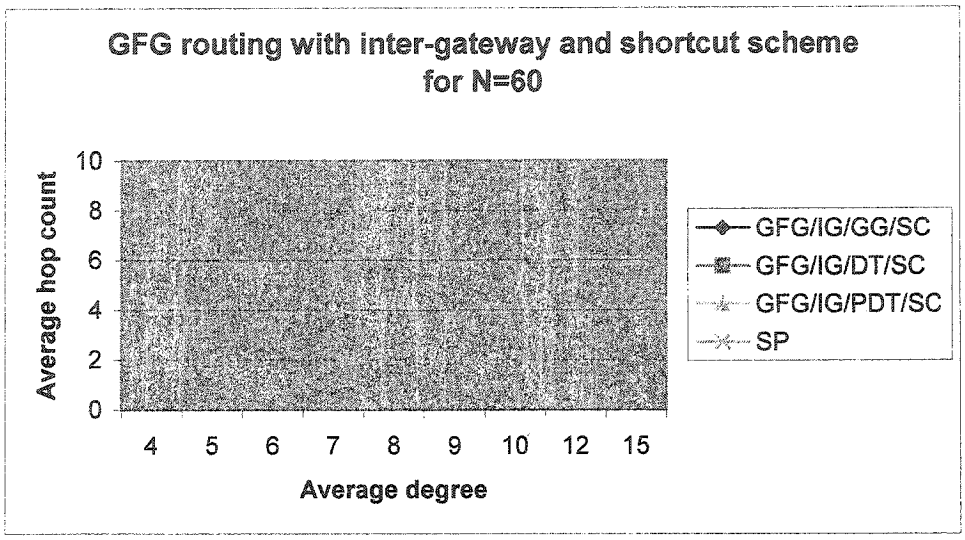


Figure 23 *GFG routing* with shortcut and *inter-gateway node* scheme

The shortcut scheme greatly reduces the average hop count, especially on the *FACE routing* scheme. For N=60 and using the *FACE routing* and PDT planarization without an *inter-gateway node* scheme, the average hop count with the shortcut scheme reduces by 33.4% on degree=4; average hop count reduces by 47.1% on degree=8; and average hop count reduces by 59.4% on degree=15. The shortcut scheme tremendously reduces the average hop count of *FACE routing* scheme. The bigger the degree, the more the effects on reducing the average hop count. The reason is that *FACE routing* adopts a snake-like path and a shortcut takes a relatively straight one. A dense graph has more possibilities to have a shortcut. The shortcut scheme also has some effects on the *GFG routing* scheme, especially with a low degree (degree < 10). Compared with a non-shortcut scheme, the *GFG routing* scheme and PDT planarization without an *inter-gateway node* scheme but when incorporated with the shortcut scheme, the average hop count reduces by 26.9% on degree=4; average hop count reduces by 6.3% on degree=8; and average hop count reduces by 1.8% on degree=15.

GFG routing scheme doesn't have an obvious effect when using a shortcut on a dense graph. On this situation, *GFG routing* scheme has a performance very close to that of the *shortest path algorithm*. The space for reducing average hop is limited when the degree is big.

The shortcut scheme doesn't have much difference among GG, DT and PDT graphs. Considering the statistic deviation, they are pretty close. This is because the shortcut scheme can take a route on an edge that is eliminated by any planarization, no matter which planarization scheme is used, if this edge exists in a UDG.

3.5 *FACE routing* with before-crossing scheme

In the fourth step, we implemented *FACE routing with before-crossing* scheme. We also used those improvements with *FACE routing* scheme in the former steps on *FACE routing with before-crossing* scheme. We don't *GFG routing* scheme incorporated with new *FACE routing* scheme because of the obvious drawbacks of the new-version FACE scheme. Figure 24 shows the results of average hop. More precise data can be seen from Table 7.

FACE routing with before-crossing scheme has a better performance than old-version *FACE routing* scheme when degree > 10 and N=60, but it has a worse performance than old-version on UDGs when degree < 10 and N=60. This is because a UDG with a small degree is more likely to have snake-like paths. New-version *FACE routing* has a back-and-forth effect on the snake-like path. This incurs longer paths between some pairs of source and destination.

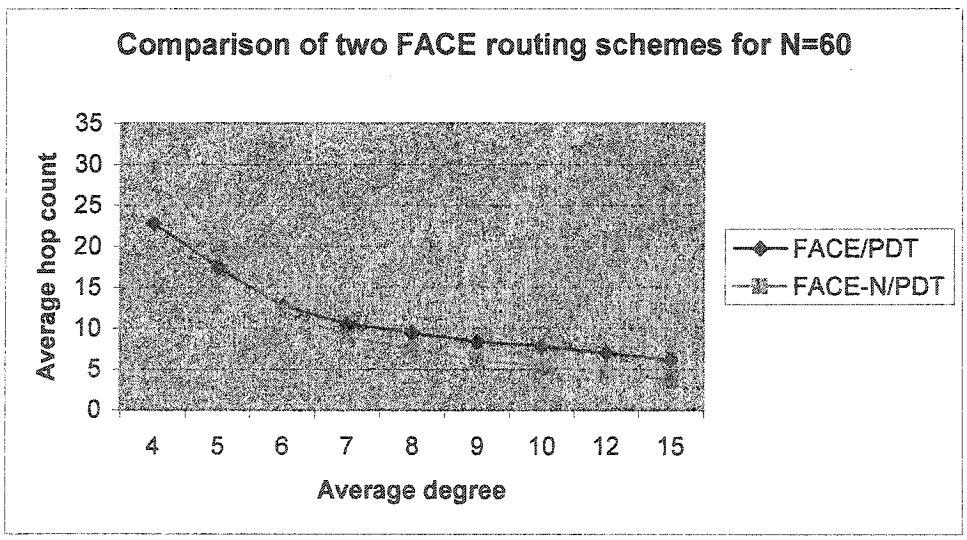


Figure 24 Comparison of two *FACE routing* schemes

In conclusion, a PDT graph is denser than a GG graph. PDT based *GFG routing* scheme has a better performance than GG based *GFG routing* scheme. The smaller the average degree is, the

less the average hop count *GFG routing* scheme has. For N=60, PDT based *GFG routing* scheme has a 9.4% smaller hop count than GG based *GFG routing* scheme when degree=4; PDT based *GFG routing* scheme has a 2.8% smaller hop count than GG based *GFG routing* scheme when degree=9; PDT based *GFG routing* scheme has a 1% smaller hop count than GG based *GFG routing* scheme when degree=15. If we use *inter-gateway node* scheme, when N=60, PDT based *GFG routing* scheme has a 2.9% smaller hop count than GG based *GFG routing* scheme when degree=4; PDT based *GFG routing* scheme and GG based *GFG routing* scheme have very close performance when degree>10. Shortcut scheme doesn't provide many improvements on PDT based *GFG routing* scheme compared with GG based *GFG routing* scheme. They are almost the same. This is because the shortcut scheme can take any path that is eliminated from the planar graph, no matter by what kind of planarization scheme is used.

Chapter 4 Address configuration

This chapter describes about the address configuration scheme. We propose to use a PDT graph instead of a GG graph to construct *home perimeters* and find *home nodes*. We also implement *inter-gateway node* scheme on address configuration. The *GFG routing* in previous chapter is used to locate the *home nodes* and *home perimeters*. We propose an outer perimeter update protocol in this chapter. We use Microsoft Visual C++ to simulate the address configuration schemes. We repeat the experiments and obtain the average and confidential interval of experiment data. The detail data for different degrees and different number of node N are present in Appendix.

4.1 Description of address configuration

Because of the hierarchical architecture of the Internet, the addressing and IP routing are strongly bound together. Here we will propose a new address configuration scheme. The new address configuration scheme uses a node on the outer perimeter of a planar graph as an *Addressing Agent*. We will present a distributed dynamic host configuration protocol designed to configure nodes in MANET.

The key issue of the *Addressing Agent* is that it acts roughly as an address configuration server. This node, acting as the *Addressing Agent*, keeps a list of all of the MANET nodes. This list contains the mapping of locations and identifiers of IP addresses.

The goal of this scheme is to provide exactly one *Addressing Agent* to each ad hoc network. If no *Addressing Agent* exists, one has to be chosen. If several *Addressing Agent*s are present in the ad hoc network, one must be chosen as the unique *Addressing Agent*. The node with the biggest x coordinate is chosen as *Addressing Agent*. This approach is interesting since it provides scenarios for uniting or splitting ad hoc networks. In the case of the union of several ad hoc networks, only the *Addressing Agent* with the biggest x coordinate is chosen.

1) Ad hoc network initiation

The selection of an *Addressing Agent* is initiated after the formation of the ad hoc network. Any node in ad hoc network can initiate the procedures for selecting an *Addressing Agent*. This node uses *GFG routing* to find the easternmost node in the ad hoc network. We consider two schemes:

one with a *dominating set* scheme and one without a *dominating set* scheme. We first consider the case without a *dominating set* scheme. GG, DT, or PDT schemes are used to extract a planar graph from the ad hoc network.. The easternmost node has the biggest x coordinate. If more than one node has the same x coordinate, we choose the one with the biggest y coordinate as the *Addressing Agent.*

To find the easternmost node, a searching packet is used to do this job. The searching packet is routed geographically to the easternmost point. The destination can be set to be infinite on the east (or any value big enough to be outside on the east of the ad hoc network). The searching packet uses two distinct modes for routing: greedy mode and FACE mode. The searching packet first uses the greedy mode scheme to forward the packet as much as possible. The greedy mode scheme fails at a local maximum point when no neighbor is closer to the destination than the local maximum point. The *GFG routing* scheme recovers from failure using the FACE mode. The searching packet continues forwarding until it encounters a node that is closer to the destination than the local maximum point. Then the *GFG routing* scheme switches back to the greedy mode. The destination can be treated as a disconnected node. When the searching packet reaches the last local maximum point it will change to the FACE mode again. The packet will tour the outer perimeter that encloses the planar graph. The last local maximum point is the *Addressing Agent.* Each node on the outer perimeter is called a replica of the *Addressing Agent.* The *Addressing Agent* knows to consume the packet when it returns after this tour of the outer perimeter.

When a node becomes the *Addressing Agent*, it will do the following things:

- Broadcast to the whole ad hoc network about its status as the *Addressing Agent*, its location, its IP and ID;
- Generate a network prefix for the network address;
- Keep the list and its sequence of nodes on the outer perimeter of the planar graph;
- Copy information about itself to nodes on the outer perimeter of the planar graph.

2) Outer perimeter update protocol (PUP)

Address configuration of ad hoc networks uses PUP to refresh the outer perimeter of the planar graph in the event of a change of network topology.

The *Addressing Agent* periodically sends PUP packets to tour the outer perimeter of the planar graph. The PUP packet will record any node and its sequence on the outer perimeter. If the *Addressing Agent* finds any change on the outer perimeter, it will copy the new collected date to all nodes on the outer perimeter so that every node on the outer perimeter can keep up-to-date information about the outer perimeter.

Each replica will store a list of other replicas and their sequence. The sequence is generated by using the *right hand rule* to tour the outer perimeter. Some nodes may appear more than one time in the sequence. Assume node A is a replica on the outer perimeter and we are only considering the edges on a planar graph. So A knows its two immediate neighbors in the sequence of the outer perimeter of the planar graph: back neighbor U and front neighbor W. Let vector $\boldsymbol{AU}$ have the angle θ_1 ($0 \leq \theta_1 < 2\pi$) with x coordinate, vector $\boldsymbol{AW}$ have the angle θ_2 ($0 \leq \theta_2 < 2\pi$) with x coordinate and $\theta_2 > \theta_1$. For any neighbor B of replica A, assume edge AB to be on the planar graph. So the angle θ of vector $\boldsymbol{AB}$ with x coordinate will not be in the range (θ_1, θ_2). Any node C comes to the communication range of node A. A detects a new neighbor C. A first checks whether edge AC is on the planar graph (GG, PDT or DT graphs). If edge AC is not on the planar graph, edge AC is not on the outer perimeter. If edge AC is on the planar graph, A calculates the angle θ_3. θ_3 is the angle of vector $\boldsymbol{AC}$ with x coordinate and $0 \leq \theta_3 < 2\pi$. If the angle θ_3 is in the range (θ_1, θ_2), A can determine that the outer perimeter of the planar graph is changed. Node A will send a new ad hoc network initialization packet to explore any new *Addressing Agent*. In other words, AW is the first edge that is covered when AU is rotated around A in a counterclockwise direction. If $\theta_2 < \theta_1$, the corresponding range is $[0, \theta_2) \cup (\theta_1, 2\pi)$.

Figure 25 shows the planar graph of a PDT and its outer boundary. Edge GH is a dangling edge. Node A, U and W are on the outer boundary. Node B is inside the planar graph. If we use the *right hand rule* to tour the outer boundary of the PDT planar graph, we will pass node G twice. For node H, its front neighbor and back neighbor on the outer boundary of the planar graph both are both G.

Each node on the outer perimeter will store such a list of nodes and their sequence of nodes on the outer perimeter. If any node detects a topology change of the outer perimeter, it will assume that a change of network happens. This node will initiate a search packet for new *Addressing*

Agent. This packet will use *GFG routing* scheme to find the new *Addressing Agent* and new outer perimeter.

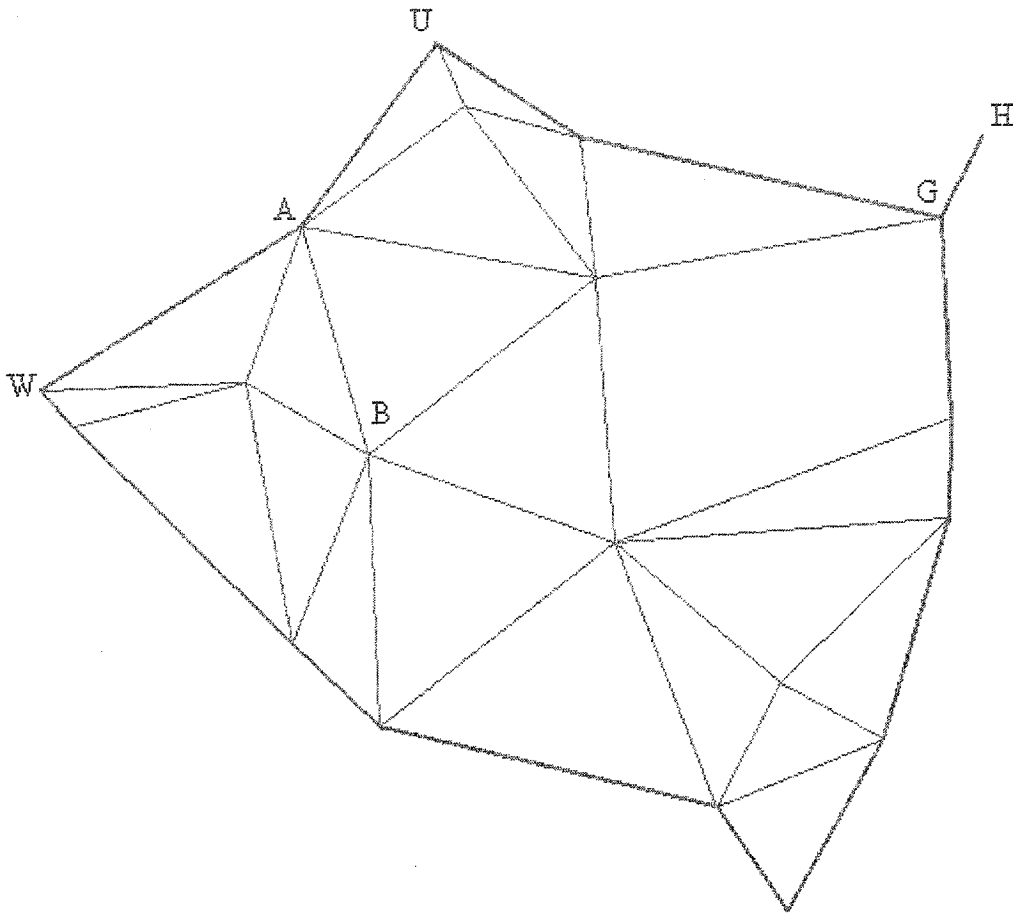


Figure 25 Outer boundary of PDT planar graph (Thick red line)

3) Node joining and leaving the MANET

An unconfigured node may wish to join the ad hoc network. The node should be configured, which includes assigning an IP address that is different from the IP addresses of all other nodes. On the other hand, a member node of the network may leave the network at any time. Some nodes may have opportunities to inform the other MANET nodes before its departure. Some nodes may abruptly depart due to node crash or network partitioning. In such a situation the remaining nodes are responsible for detecting the departure. The *Addressing Agent* and its replicas will be in charge of both of these two cases and reclaim the IP addresses of the departing nodes.

If a node, *X*, requests an IP address, it will first explore whether there is a neighbor that is a replica on the outer perimeter. When *X* has a replica neighbor, *Y*. *X* will directly asks for an IP

address from *Y* instead of the *Addressing Agent*. *Y* gives an IP address to *X*. *Y* uses PUP protocol to keep data consistent with other replicas and the *Addressing Agent*. If *X* doesn't have such a replica neighbor, *X* will send an address request to the *Addressing Agent*. The *Addressing Agent* gives an IP to *X*. The *Addressing Agent* also uses PUP protocol to keep data consistent with replicas.

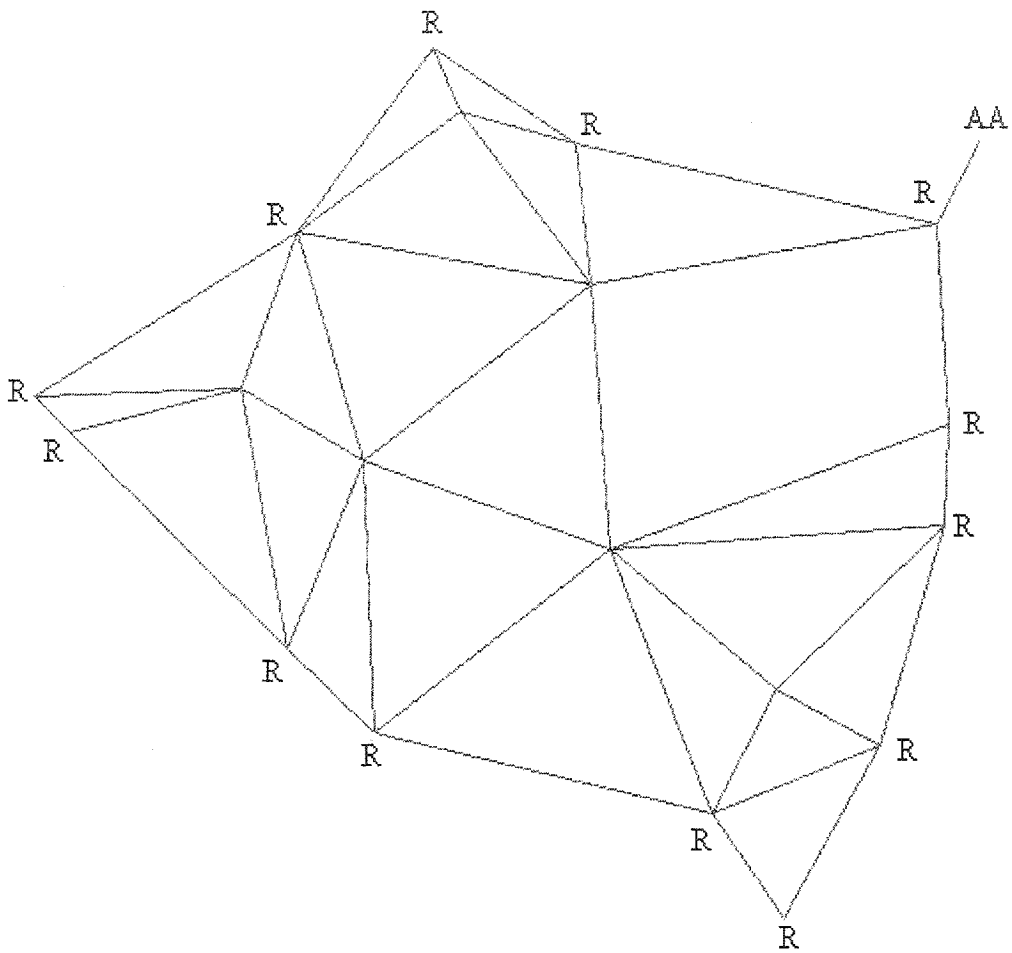


Figure 26 The *Addressing Agent* (AA) and replicas (R) on the PDT planar graph

4) Network partitioning and merger

At any time of operation, a given ad hoc network may be split into multiple partitions. Subsequently, partitions may also merge. Partitioning of the network is comparatively easy to handle. When an ad hoc network splits into different partitions, there won't be any address conflicts in either of the resulting partitions; but only one of them will own the old *Addressing Agent*. The other partitions will have to elect a new *Addressing Agent* in a common way. The procedure of ad hoc network initiation is used to find the new *Addressing Agent*s for these partitions.

In the case of the union of several ad hoc networks, the new network will choose a new *Addressing Agent*. The procedure is still the same. The easternmost node in the new network is defined as the *Addressing Agent*. PUP is used to refresh the new outer perimeter. Some nodes will have to be readdressed and the communication will be interrupted within them. But after the readdressing, the new ad hoc network will be a coherent set of configured nodes.

5) *Addressing Agent* with *inter-gateway* scheme

The *inter-gateway nodes* are a sub set of the total nodes in ad hoc networks. The *Inter-gateway node* scheme may reduce the number of nodes in a network. We can use *inter-gateway node* to have a smaller connected network. Then the easternmost node on the *inter-gateway node* sub graph is chosen as the *Addressing Agent*. We extract a planar graph from the sub graph. Each node on the outer perimeter of *inter-gateway node* sub planar graph is a replica of the *Addressing Agent*. This approach can reduce the number of total replicas. Figure 27 shows the *Addressing Agent* and its replicas with the *inter-gateway* scheme. The size of the outer perimeter of a PDT of *inter-gateway node* sub graph is smaller than that of a PDT without *inter-gateway node* scheme.

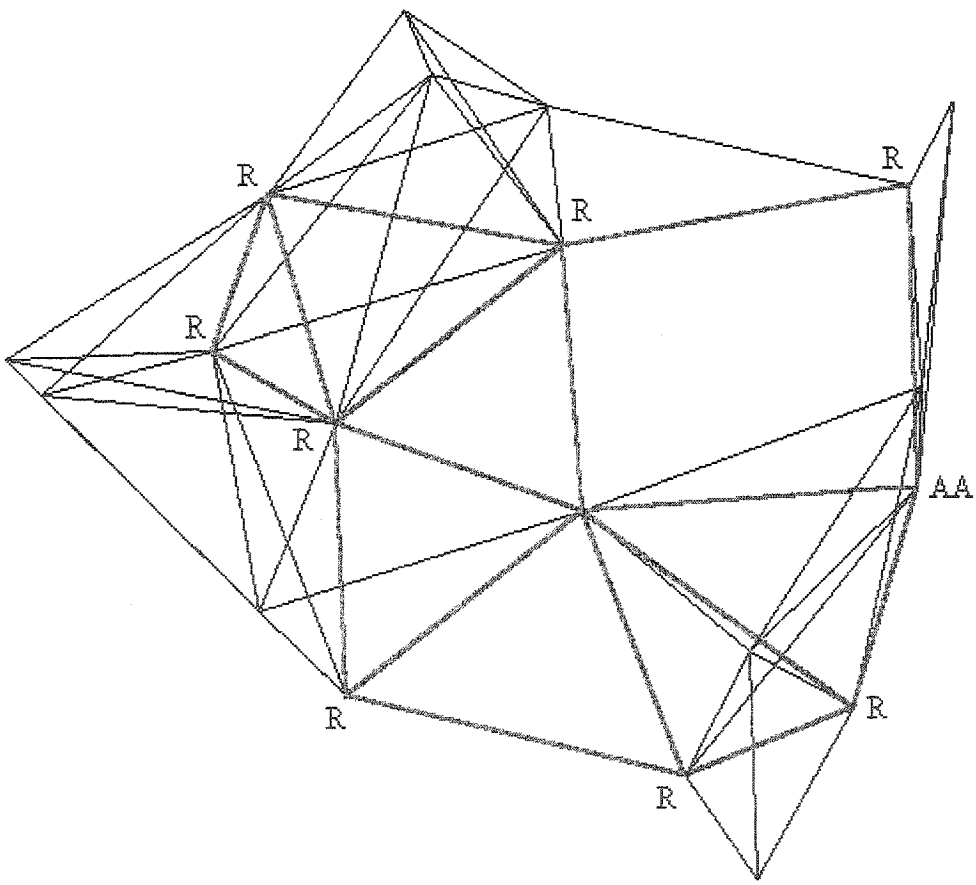


Figure 27 The *Addressing Agent* and replicas with *inter-gateway node* and PDT scheme (Red: *inter-gateway node* sub planar graph)

4.2 Duplicate Address Detection (DAD) using GHT

It is a major problem in ad hoc networks to realize DAD. An ad hoc network doesn't have a central administration. When a node requests an IP address, it is important for the node to check whether other nodes have already used this address. DAD is a procedure that ensures two nodes in the same network can't be configured with the same IP address. Because of the hierarchical architecture of the Internet, IP addresses have a prefix in any subnet. When two ad hoc networks merge, there is possibility that these two subnets have the same prefix for addresses. So, duplicate IP addresses will probably exist in the combined network. DAD is used to settle this problem. We propose using GHT to detect duplicate IP addresses. This method uses *GFG routing* scheme to send duplicate address verification messages. The GHT decides where the *home node* and *home perimeter* are.

1) *Home node* and *home perimeter*

The core step in GHT is the hashing of an IP address into geographic coordinates. For a node *A*, which has the IP address *X*, GHT hashes *X* pointing to a position in the network. This position is unique. No matter which node has this IP address, the mapped position is the same place. The hashed point could be a node in the network, but more likely it is an isolated point without any connection to the network. *GFG routing* scheme is used to forward duplicate address verification packets to the hashed points. Because the hashed point can be seen as a disconnected point from the network, when we use *GFG routing* scheme to locate this point, the packet will traverse the entire FACE that encloses the destination. This FACE is the *home perimeter*. The nodes on the *home perimeter* are called replicas that copy address information from the *home node*.

There is the possibility that GHT will hash an IP address to a point outside the network. When a packet gets to the node that is closest to the hashed point, it will change to FACE mode. The packet will go around the outer perimeter. Under this circumstance, the *home node* should be on the outer perimeter of the planar graph. The node that is closest to the hashed point on the outer perimeter is the *home node*. The outer perimeter of the planar graph is the *home perimeter*. Each node on the outer perimeter is a replica of the *Addressing Agent*. Communication overheads are needed among all the nodes on the outer perimeter to keep data consistent and persistent between the *home node* and its replicas. This case may not be a good one because the outer perimeter has a big percentage of the nodes. The communication overheads between the *home node* and its

replicas are proportional to the number of nodes. A small perimeter is preferable because it reduces communication overheads. The selection of hash function should avoid such a situation.

2) Duplicate address verification

All nodes send their IP addresses to the hashed points to verify duplicate IP addresses. The operation is fully distributed. The hashed points spread across the whole network. *GFG routing* scheme uses a single path strategy without any memorization. Hence the network doesn't need broadcasting. This approach also has good scalability. The selection of the *home node* is localized no matter what kind of size the network is. In the case of a network merger, two nodes with the same IP address will send verification packets to the same point. *GFG routing* scheme will forward packets to the destination. In this way duplicate address detecting can guarantee success. In the case of splitting, although two nodes in the separated networks may have the same IP address and the hashed points will be the same if they use the same hash function, the *home node*s for this IP address in the two different partitions will be different. Each *home node* will be in charge of checking duplicate IP addresses within its own subnet. In this way, two nodes with the same IP address on different networks will not have a conflict.

If two nodes in the same ad hoc network have the same IP address, they both send duplicate address verification packets to the same-hashed location. When the *home node* or any replica receives two packets inquiring for the same IP address, it can perceive that duplicate IP address exists. It sends *NAK* information to the later node for this IP address. This node receives this packet and knows that the other nodes already use this IP address. So it sends an IP request to the *Addressing Agent* for a new address. This DAD scheme doesn't use any central node at all. The communication overheads happen locally within the network if we choose a good hash function. The method can guarantee success because *GFG routing* scheme can guarantee delivery.

4.3 Advantages and drawbacks of the address configuration scheme

The address configuration scheme for ad hoc networks has the following advantages:

- Localized manner: This method only uses position-based routing. Any ad hoc network can choose a unique *Addressing Agent* within a network using localized location information

- Single path strategy: Outer perimeter based address configuration uses the single path strategy. It avoids the resorting of flooding.
- The *Addressing Agent* is on the easternmost point of the network. So each node in the network can assume it knows where the *Addressing Agent* is.
- Scalability: The proposed scheme handles merger and splitting easily, no matter what kind of size the network is.
- Guaranteed success: the *GFG routing* scheme can guarantee to find the *Addressing Agent* and the *home perimeter* of planar graph.
- Distributed duplicate address detection: GHT is used to detect duplicate address. The hashed points spread across the whole network.

This address configuration scheme has the following drawbacks:

- The search packet may take a long path to find the *Addressing Agent* because the *Addressing Agent* is on the easternmost point of the network.
- Similarly, because the *Addressing Agent* is on the easternmost point of the network, it will have bigger communication overheads to broadcast information to all nodes in the network.
- If any new node with a bigger x coordinate than the *Addressing Agent* joins the network, the network will change the *Addressing Agent*.
- For a sparse network, the nodes on the outer perimeter of a planar graph may have a relatively big percentage of the total nodes. For example, when N=60 and d=4, where N is the number of total nodes and d is the average degree of the UDG, the percentage of nodes on outer perimeter of PDT is 89.2%.
- For a very dense network, if we use the *inter-gateway* node sub set to find the *Addressing Agent*, there may be the problem that there isn't any *inter-gateway* node.
- Each node using position-based routing needs a GPS device. This equipment is not cheap and GPS has limitations on accuracy. The GPS signal also can be shielded when devices are indoors.

4.4 An example of the address configuration protocol

Here we present an example of an address configuration scheme. This address configuration scheme is fully distributed. There is no central administration in this scheme. *GFG routing* scheme and GHT are used for DAD. Following text is the detailed description:

First, let node *A* be the first node in an ad hoc network. Node *A* initiates the formation of the ad hoc network. *A* will determine the prefix of the IP address. *A* randomly chooses an IP address for itself (with the determined prefix).

Second, node *B* hopes to join the network. Node *B* asks for an IP address prefix from any configured node in the network. At this time, the network has only one node *A*. So *B* will ask node *A* for an address prefix. When node *B* gets a prefix, it randomly chooses an IP address, *X* (with the determined prefix). Node *B* uses GHT to hash *X* to a point in the network. Then node *B* uses *GFG routing* to forward a duplicate address verification packet to the hashed point. The *home node* for this hashed point will be in charge of checking duplicate IP addresses. At this point, the only node *A* is the *home node* for this IP address, *X*.

Third, a node, *C*, hopes to join the network. It asks for an address prefix from any node in the network. When node *C* gets the address prefix, it will randomly choose an IP address *Y* (with the determined prefix). Node *C* uses GHT to hash the IP address, *Y*, to a point within the network. Then node *C* uses *GFG routing* scheme to forward a duplicate address verification packet to the *home node* of this point. The home node is in charge of checking duplicate addresses. If no other node uses this IP address, *Y*, the *home node* will send back an *ACK* packet for this IP address, *Y*. Otherwise, the *home node* will send back a *NAK* packet for this IP address. When node *C* receives *ACK*, it knows that no other node uses the IP address *Y*. Node *C* keeps the IP address, *Y* for itself. When node *C* receives *NAK*, it knows that another node already has the IP address, *Y*. Node *C* will randomly choose another IP address, *Z*. The above procedures repeat until node *C* gets a unique address.

This scheme for address configuration is totally distributed. There is no central administration in the network. Any node in the network can act as the *home node*. This scheme is scalable and handles network partitioning and merger easily.

One big issue here is how to cope with the change of *home node*. *Home node*s may be inactive or leave the network. The topology of a network may change. All of these will incur the change of some *home node*s. It is important to keep information consistent between old *home node* and new *home node* of the hashed point. Some communication overheads are needed for this situation.

4.5 Performance evaluation

In the experiments, we find the outer perimeter of the planar graph using GG, PDT, and DT planarization schemes. The *inter-gateway node* scheme is also used to generate a sub graph. The size of the outer perimeter of *inter-gateway node* sub planar graph is also measured. Figure 28 shows the average size for N=60. More precise data can be seen from Table 9. Each case has 30 samples.

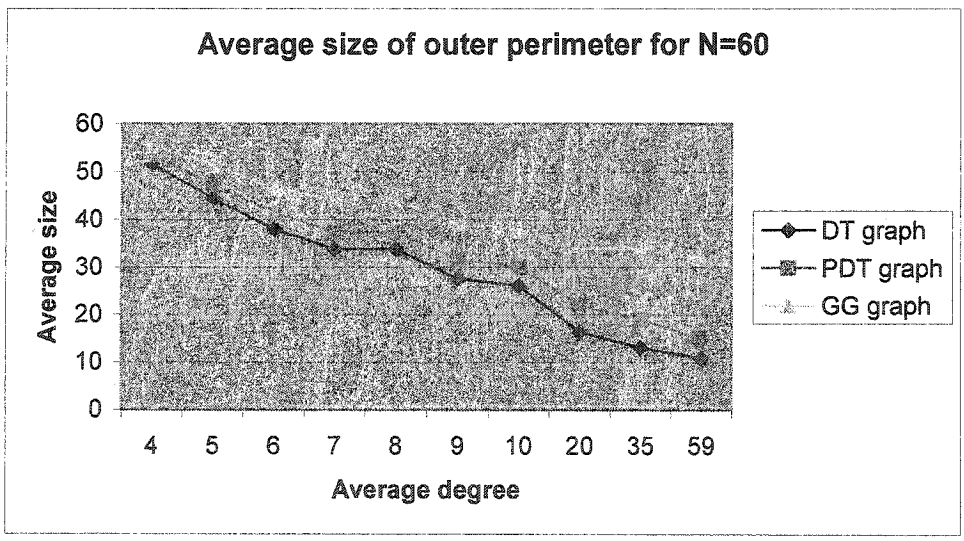


Figure 28 Average size of outer perimeter on different planar graphs

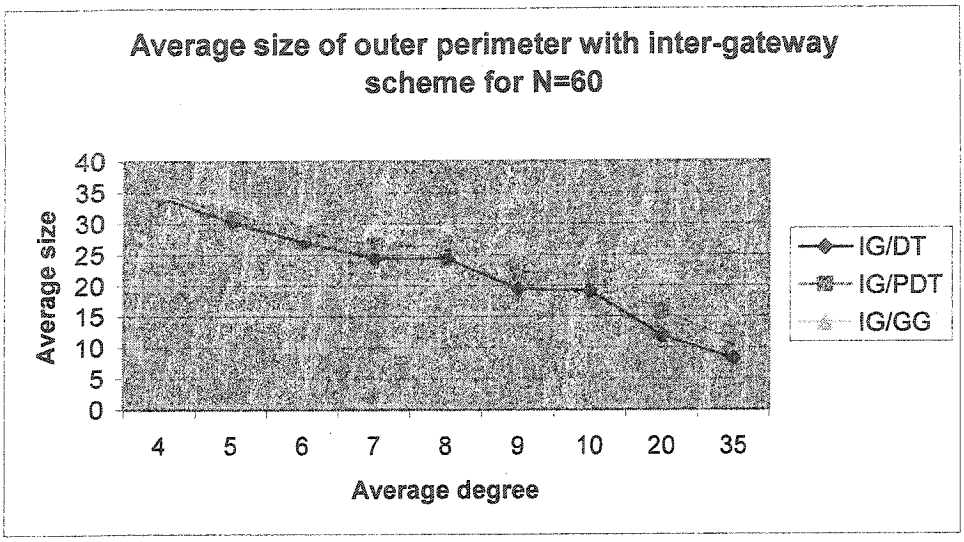


Figure 29 Average size of outer perimeter with *inter-gateway node* scheme

From Figure 28 we can see that a PDT graph has less nodes on the outer perimeter than a GG graph, especially on dense networks. For N=60 and d=5, when compared to a GG graph, a PDT graph has 15.1% fewer nodes on the outer perimeter than a GG graph; for N=60 and d=10, a PDT graph has 26.3% fewer nodes on the outer perimeter than a GG graph; for N=60 and d=20, a PDT graph has 38% fewer nodes on the outer perimeter than a GG graph. The advantage of the PDT is obvious.

It can be seen from Figure 29 that the *inter-gateway node* scheme has a smaller set of nodes, but the *inter-gateway node* may have problems on very dense networks. This will happen when no *inter-gateway node* exists in the network.

5. Data-centric storage in sensornets

In this chapter, we propose to use a PDT graph for data-centric storage. The PDT graph is used to construct a planar graph. The *home perimeters* and *home nodes* for data-centric storage are found on the constructed planar graph. We also use GG and DT graph in the experiments. We use Microsoft Visual C++ to implement the protocols. We repeat the experiments and obtain the average and confidential interval. The detail data for different number of node N and different degrees are present in the Appendix.

5.1 Data-centric Storage

In DCS, relevant data are stored by name at nodes within the sensornet. DCS uses GHT to hash names into geographic coordinates. The *GFG routing* scheme is used to store and retrieve data. In this paper, we propose an improvement on DCS. This improvement uses the *GFG routing* scheme and the PDT graph. They are used to find *home nodes* and *home perimeters*.

In DCS, most data traffic happens among the *home perimeters*. So the size of *home perimeters* will affect the performance of DCS. Different planarization schemes may have different sizes of *home perimeters*. We use GG, DT, and PDT to construct planarization graph. We first use a UDG to represent sensornets. Then, these different planarization schemes are used to generate planar graphs.

Authors [HLR] also presented some statistics on Gabriel graphs, which they used in the algorithm as a planar graph. On a random graph with 1600 nodes, they report average FACE size of 4.3, 1369 FACEs, average node degree about 4, and the average size of outer FACE of 248. However, they counted each dangling edge (edge that belongs to only one FACE, not to two different FACEs). We believe that this is not a correct count since in FACE traversals, in all applications, these edges are indeed visited twice, not once, therefore each dangling edge shall be counted as two edges in the corresponding FACE, not as only one edge. With this change, counting average number of edges on a FACE of a planar graph becomes quite easier. Let N, F, and E be number of nodes, FACEs, and edges of a planar graph, respectively. The well-known Euler formula is $F=E-N+2$. Let S be average number of edges on a FACE. Since each edge is counted twice (whether or not it belongs to same or two different FACEs), we have $FS=2E$. Thus

$S=2E/F$, or $S=2E/(E-N+2)$. We use this new algorithm to do experiment. The results are present in the thesis.

5.2 Performance evaluation

In the experiments, we used different planarization schemes on sensornet networks. We used GG, DT, and PDT to build planar graphs. We found the average size of all *home perimeters* in a planar graph. Figure 30 shows the average number of hops if we use the *right hand rule* to tour the *home perimeters*. More precise data can be seen from Table 15. There are 30 samples for each case. We compute the average hop count

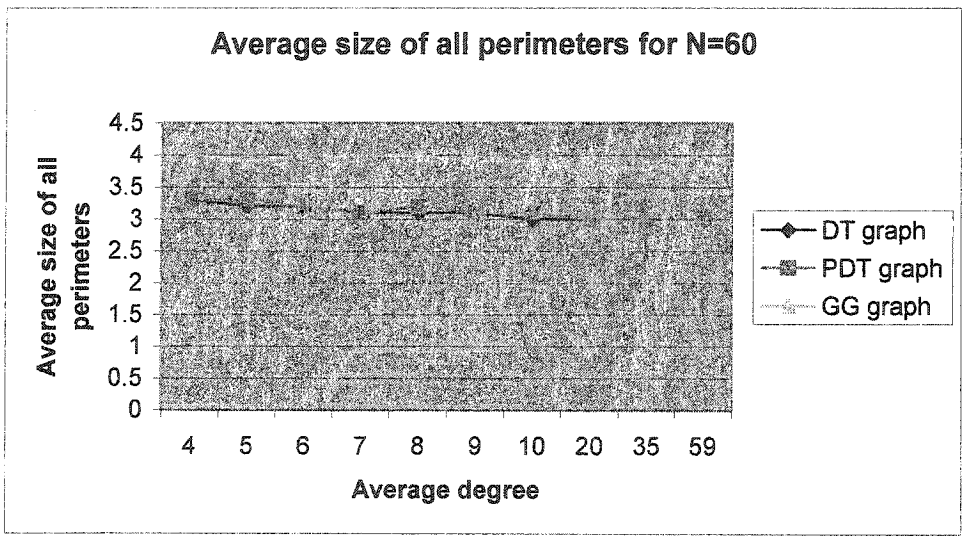


Figure 30 Average size of all perimeters on different planar graph

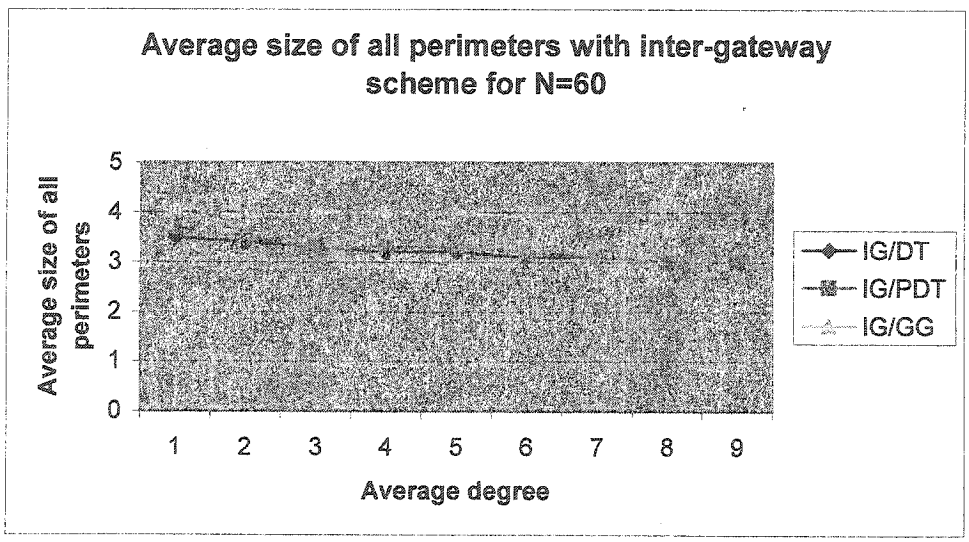


Figure 31 Average size of all perimeters with inter-gateway scheme

It can be seen that DT and PDT graphs have smaller perimeters than GG graphs. A DT graph can't be constructed in a localized manner, but a PDT graph can be constructed in a localized manner. DT graph and PDT graph have very close performance on average sizes of perimeters. So a PDT graph can be used for finding *home perimeters*. When N=60 and d=4, the average size of a PDT-based perimeter is about 17.5% less than the size of a GG-based perimeter. When N=60 and d=35, the average size of the PDT-based perimeters is about 21% less than that of a GG-based perimeters. A smaller perimeter means fewer hops on the *home perimeter*. So there will be less communication overheads on PDT-based data centric storage.

We notice that the *inter-gateway node* sub graph doesn't change the average size of perimeters very much compared to its mother graph. The *Inter-gateway node* method may reduce the number of total nodes and thus reduce the number of *home perimeters*.

Chapter 6. Conclusion

This chapter summarizes conclusions of former chapters and gives the directions of future researches.

6.1 Main contribution of this thesis

This paper proposes to use a PDT graph on the *FACE routing* scheme and the *GFG routing* scheme. The two routing schemes are position-based routing algorithms and can guarantee delivery. A PDT graph can be constructed in a localized manner. It has been shown that PDT graphs have better performance than the previous used GG graphs. A PDT graph incorporated with a *dominating set* further enhances the performance of these two routing schemes.

In this thesis, we proposed schemes for address configuration for ad hoc networks. A node with the biggest x coordinate is chosen as the *Addressing Agent*. GG, DT, or PDT schemes are used to extract a planar graph from ad hoc networks. Nodes on the outer perimeter of a planar act as replicas of the *Addressing Agent*. The *GFG routing* scheme is used to find the *Addressing Agent* and the outer perimeter of the planar graph. The *GFG routing* algorithm is also used for network initiation and perimeter update. The new approach can easily handle network splitting and merger. It only uses a single path strategy and can guarantee success. Our experiments show that the new approach performs better on PDT graphs than on GG graphs. We propose to use GHT for duplicate address detection. In this scheme, each node sends its IP verification packet to a hashed point. A node on the *home perimeter* is in charge of detecting duplicate IP addresses. This scheme is fully distributed. It is scalable and can handle network splitting and merger easily. We provided an example of an address configuration scenario where each node sends duplicate IP address verification packets to a hashed point. A home node that is nearest to the hashed point is in charge of checking the IP address. This scheme is fully distributed.

In this thesis, we use the *GFG routing* algorithm and PDT planarization schemes on DCS. The *GFG routing* scheme is used to find the *home perimeters* and the *home nodes*. The results show that PDT graphs have better performance on DCS than on GG graphs. A detailed algorithm on how to get average size of *home perimeter* is given in this thesis.

Routing, address configuration and data-centric storage are relevant with each other in this thesis. We use the *GFG routing* to deliver packages. We also use the *GFG routing* to construct *home perimeters* and find *home nodes* for address configuration and data-centric storage. Address configuration provides IP addresses for data-centric storage. The PDT graph is used in all these three areas for planar graphs. The experiment results show the advantages of the PDT graph scheme.

6.2 Future research

It may be an interesting topic using power-aware routing scheme on sensornets. In this thesis, we don't consider power consumption when we use a routing algorithm to find the paths between sources and destinations. The shortest path may be the best choice for one route, but it is not the best solution for the whole networks. Power is a scarce commodity in sensornets. It may not be the best choice to let traffic concentrate on a few nodes. This will quickly exhaust the energy reserves of these nodes. A path that consumes less energy could be a better choice for sensornets although it may take a longer path. So the power-aware and power-efficient routing algorithm may be a better choice for sensornet routing. A routing scheme should consider energy consumption. And good routing scheme should minimize the total energy per packet and/or lifetime of each node.

Another interesting topic for future work could be how to simulate sensornets using a better simulation tool. In our research, we used the UDG graph to model sensornets. We constructed planar graphs on UDG. In the real world, the network may be in 3-D space, there may be obstacles between nodes, nodes may have unequal transmission powers and networks may have indirect links. All of these may affect the paths between any two nodes. A UDG graph cannot describe these things. Finding a better model to simulate sensornets while overcoming these drawbacks is a challenging work.

References

[BCSW] S. Basagni, I. Chlamtac, V. R. Syrotink, B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," Proc. MOBICOM, 1998, pp. 76-84.

[BGL] Lj, Blazevic, S. Giordano and J. Y. LeBoudec, "Self organized terminode routing," TR DSC/2000/040, Swiss Federal Institute of Technology, Lausanne, December 2000.

[BHSM] Milind Buddhikot, et al, "Mobile NAT: A New Technique for Mobility Across heterogeneous Address Spaces," WMASH'03, San Diego, California, USA, September 19, 2003.

[BMSU] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," 3nd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Seattle, August 20, 1999, pp. 48-55; Telecommunication System, to appear.

[DSW] S. Datta, I. Stojmenovic and J. Wu, "Internal Nodes and shortcut Based Routing with Guaranteed Delivery in Wireless Networks," Cluster Computing.

[F] G. G. Fin, "Routing and Addressing problems in large metropolitan-scale internetworks," ISI Research Report, ISU/RR-87-180, March 1987.

[GR] M. Gunes and J. Reibel, "An IP Address configuration algorithm for Zeroconf. Mobile Multi-hop Ad Hoc Networks," In Proceedings of the International Workshop on Broadband wireless Ad-Hoc Networks and Services, Sophia Antipolis, France, September 2002.

[GS] K. Gabriel and R. Sokal, "A new statistical approach to geographic variation analysis," Systematic Zoology, 18, 1969, pp. 259-278.

[HL] T.C. Hou and V.O. K. Li, "Transmission range control in multihop packet radio networks," IEEE Transaction on Communications," 34, 1, 1886, pp. 38-44.

[HLR] Q. Huang, C. Lu, G.C. Roman, "Reliable mobicast via face-aware routing," IEEE INFOCOM 2004.

[HP] Z. Haas and M. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," ACM/IEEE Trans. Net. Vol. 9, No. 4, Aug. 2001, pp. 427-438.

[JM] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," Mobile Computing, ed. T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 153-181.

[JPS] R. Jain, A. Puri and R. Senngupta, "Geographical Routing Using Partial Information for Wireless Ad hoc networks," IEEE Personal. Comm., Feb. 2001, pp. 48-57.

[KK] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," Proc. MOBICOM, August 2000, pp. 243-254.

[KSEGYY] Brad Karp, et al., "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," Mobile Networks and Application, August 2003, pp. 427-442.

[KSU] E. Kranakis, H. Singh and J. Urrutia, "Compass Routing on Geometric Networks," Proc. 11th Canadian Conf. On Computational Geometry, Aug. 1999.

[KV] Y. B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," MOBICOM, 1998, pp. 66-75; Wireless Networks, 6, 4, July 2000, pp. 307-321.

[KW] F. Kuhn, R. Wattenhofer, A. Zollinger, "Ad-hoc networks beyond unit disk graphs," ACM DIALM-POMC, San Diego, Sept. 2003, pp. 69-78.

[L] J, Li et al. "A Scalable Location Service for Geographic Ad Hoc Routing," Proc. ACM MOBICOM 2000, pp. 120-30.

[LCWW] X.Y. Li, G. Calinescu, P.J. Wan and Y. Wang, "Localized Delaunay triangulation with application in ad hoc wireless networks," IEEE Transactions on Parallel and Distribution Systems, Oct. 2003, pp. 1035-1047.

[LMS] X.Y. Li, K. Moaveninejad, W.Z. Song, "Robust position-based routing for wireless ad hoc networks," WWAN at IEEE ICDCS, Tokyo, March 2004, to appear.

[LSW] X. Y. Li, I. Stojmenovic and Y. Wang, "Localized scatternet formation to enable Bluetooth-based ad hoc networks," manuscript, June 2001.

[LTS} W. H. Liao, Y. C. Tseng, J. P. Sheu, GRID, "A fully location-aware routing protocols for mobile ad hoc networks," Proc. IEEE HICSS, January 2000.

[MC] J. P. Macker and M. S. Corson, "Mobile ad hoc networking and the IETF," Mobile Computing and Communication Review, 2, 1, 1998, pp. 9-14.

[MG] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," ACM Mobile Networks and Applications Journal, Special Issue on Routing in Mobile Communication Networks, October 1996, pp. 183-197.

[MW] Martin Mauve and Jorg Widmer, "A Survey on Position-Based Routing in Mobile Ad Hoc Networks," IEEE Network, Nov/Dec 2001, pp. 30-39.

[N] Nokia Rooftop wireless routing system, http://www.nokia.com.

[NP] Sanket Nesargi, Ravi Prakash, "MANETconf: configuration of Hosts in a Mobile Ad Hoc Network," IEEE INFOCOM, 2002.

[OC] Jean-Marie Orset and Ana Cavalli, "Secure Hosts Autoconfiguration in Mobile Ad Hoc Networks," IEEE INFOCOM, 2003.

[OMA] C.N. Ojeda-Guerra, D. Marrero, I. Alonso-Gonzalez, "A new approach to host configuration in an ad hoc wireless network based on DHCP," Proc. IASTED Par. Distr. Computing and Networks, Innsbruck, Austria, February 2004.

[PB] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," Computer Communications Review, October 1994, pp. 234-244.

[PC] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," Proceedings of INFOCOM'97, April 1997.

[PR] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications, February 1999.

[RF] Julien Ridoux and Anne Fladenmuller, "Address Configuration Scheme in Mobile Ad Hoc Networks," Proc. IFIP MedHoc, Tunisia, June 2003.

[RRPSS] A. Rao, S. Rathasamy, C. Papadimitriou, S. Shenker, I. Stoica, "Geographic routing without location information," Proc. ACM MOBICOM, 2003, pp. 96-108.

[S] Ivan Stojmenovic, "Position-Based Routing in Ad Hoc Networks," IEEE Communication Magazine, July 2002, pp. 2-8.

[SGS] M. Seddigh, J, Solano Gonzalez, and I. Stojmenovic, "RNG and Internal Node Based Broadcasting Algorithm for Wireless One-to-One networks," ACM Mobile Computing and Communication Rev., vol. 5, No. 2, 2001.

[SL] I. Stojmenovic and Xu Lin, "Loop-free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks," IEEE Trans, Parallel Distribution System, Vol. 12, no. 10, 2001, pp.1023-1032.

[SK] L. Subramnian and R. H. Katz, "An Architecture for Building Self-Configurable Systems," IEEE 2002.

[TK] H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals," IEEE Trans. Commun., Vol. 32, No. 3, 1984, pp. 246-57.

[V] Nitin H. Vaidya, "Weak duplicate Address Detection in Mobile Ad hoc Networks," MOBIHOC'02, EPFL Lausanne, Switzerland, June 9-11, 2002.

[WL] W. Peng and X. C. Lu, "On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks," Proc. First Ann. Workshop Mobile and Ad Hoc Networks and Computing, Aug. 2000, pp. 129-130.

[ZNM] Hongbo Zhou, Lionel M. Ni and Matt W. Mutka, "Prophet Address Allocation for Large Scale MANETs," IEEE INFOCOM 2003.

Acronym

AA: Addressing Agent

ACK: Acknowledgment

AODV: Ad hoc On-Demand Distance-Vector

CDS: connected dominating sets

DAD: Duplicate Address Detection

DCS: Data-Centric Storage

DHCP: Dynamic Host Configuration Protocol

DSDV: Destination-Sequence Distance Vector

DSR: Dynamic Source Routing

DT: Delaunay triangulation

FACE: FACE routing

FACE/DT: FACE routing on Delaunay triangulation planar graph

FACE/GG: FACE routing on Gabriel planar graph

FACE/PDT: FACE routing on partial Delaunay triangulation planar graph

FACE/IG/DT: FACE routing with inter-gateway on Delaunay planar graph

FACE/IG/GG: FACE routing with inter-gateway on Gabriel planar graph

FACE/IG/PDT: FACE routing with inter-gateway on partial Delaunay graph

FACE/GG/SC: FACE routing with shortcut on Gabriel graph

FACE/DT/SC: FACE routing with shortcut on Delaunay triangulation graph

FACE/PDT/SC: FACE routing with shortcut on partial Delaunay triangulation graph

FACE/IG/GG/SC: FACE routing with inter-gateway and shortcut on Gabriel graph

FACE/IG/DT/SC: FACE routing with inter-gateway and shortcut on Delaunay triangulation graph

FACE/IG/PDT/SC: FACE routing with inter-gateway and shortcut on partial Delaunay triangulation graph

FACE-N: second FACE routing

FACE-N/DT: second FACE routing on Delaunay triangulation planar graph

FACE-N/GG: second FACE routing on Gabriel planar graph

FACE-N/PDT: second FACE routing on partial Delaunay triangulation planar graph

FACE-N/IG/DT: second FACE routing with inter-gateway on Delaunay planar graph

FACE-N/IG/GG: second FACE routing with inter-gateway on Gabriel planar graph

FACE-N/IG/PDT: second FACE routing with inter-gateway on partial Delaunay graph

FACE-N/GG/SC: second FACE routing with shortcut on Gabriel graph

FACE-N/DT/SC: second FACE routing with shortcut on Delaunay triangulation graph

FACE-N/PDT/SC: second FACE routing with shortcut on partial Delaunay triangulation graph

FACE-N/IG/GG/SC: second FACE routing with inter-gateway and shortcut on Gabriel graph

FACE-N/IG/DT/SC: second FACE routing with inter-gateway and shortcut on Delaunay triangulation graph

FACE-N/IG/PDT/SC: second FACE routing with inter-gateway and shortcut on partial Delaunay triangulation graph

GEDIR: Geographic Distance Routing

GFG: Greedy-Face-Greedy routing

GFG/DT: GFG routing on Delaunay triangulation planar graph

GFG/GG: GFG routing on Gabriel planar graph

GFG/PDT: GFG routing on partial Delaunay triangulation planar graph

GFG/GG/SC: GFG routing with shortcut on Gabriel graph

GFG/DT/SC: GFG routing with shortcut on Delaunay triangulation

GFG/PDT/SC: GFG routing with shortcut on partial Delaunay triangulation

GFG/IG/GG/SC: GFG routing with inter-gateway and shortcut on Gabriel graph

GFG/IG/DT/SC: GFG routing with inter-gateway and shortcut on Delaunay triangulation

GFG/IG/PDT/SC: GFG routing with inter-gateway and shortcut on partial Delaunay triangulation

GG: Gabriel Graph

GHT: Geographic Hash Table

GPS: Global Position System

IG: inter-gateway

LAR: Location Aided Routing

MAC: Media Access Control

MANET: mobile ad hoc network

MFR: Most Forward within Radius

MIP: Mobile IP

NAK: Non-Acknowledgment

NFP: Nearest with Forward progress

PDT: Partial Delaunay triangulation

PRP: perimeter refresh protocol

PUP: Outer perimeter update protocol

RPM: Random progress method

SC: shortcut

TORA: Temporally-Ordered Routing Algorithm

UDG: Unit Disk Graph

WLAN: Wireless Local Areas Networks

WRP: Wireless Routing Protocol

Appendix A Experiment date of *FACE routing* and *GFG routing* for N=60

Table 1 Average hop count and 95% CI for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/GG	23.42±1.30	18.17±1.32	13.06±1.04	11.01±0.89	9.71±0.65
FACE/DT	22.13±1.18	16.84±1.12	12.34±0.76	10.22±0.63	9.16±0.40
FACE/PDT	22.79±1.31	17.42±1.19	12.74±0.83	10.52±0.69	9.42±0.46
GFG/GG	15.47±1.55	9.75±1.15	6.36±0.75	5.09±0.71	4.04±0.43
GFG/DT	13.76±1.36	8.61±0.92	5.82±0.58	4.66±0.55	3.76±0.27
GFG/PDT	14.15±1.40	8.98±1.00	5.93±0.61	4.73±0.58	3.83±0.31
SP	6.63±0.34	5.19±0.24	4.29±0.16	3.82±0.17	3.43±0.09

Table 1 Average hop count and 95% CI for *N*=60 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/GG	8.41±0.44	7.68±0.38	6.72±0.32	5.811±0.175
FACE/DT	8.19±0.26	7.61±0.19	6.84±0.15	6.054±0.089
FACE/PDT	8.37±0.31	7.81±0.25	6.96±0.20	6.114±0.103
GFG/GG	3.40±0.16	3.05±0.09	2.67±0.11	2.273±0.037
GFG/DT	3.30±0.13	2.97±0.06	2.64±0.08	2.256±0.020
GFG/PDT	3.32±0.15	2.98±0.07	2.64±0.08	2.258±0.021
SP	3.12±0.04	2.89±0.04	2.59±0.03	2.247±0.018

Table 2 Average dilation and 95% CI for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/GG	3.53±0.20	3.50±0.25	3.04±0.24	2.88±0.23	2.83±0.19
FACE/DT	3.34±0.18	3.24±0.22	2.88±0.18	2.68±0.16	2.67±0.12
FACE/PDT	3.44±0.20	3.36±0.23	2.97±0.19	2.75±0.18	2.75±0.13
GFG/GG	2.33±0.23	1.88±0.22	1.48±0.17	1.33±0.19	1.18±0.13
GFG/DT	2.08±0.21	1.66±0.18	1.36±0.14	1.22±0.14	1.10±0.08
GFG/PDT	2.13±0.21	1.73±0.19	1.39±0.14	1.24±0.15	1.12±0.09

Table 2 Average dilation and 95% CI for *N*=60 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/GG	2.70±0.14	2.66±0.13	2.59±0.12	2.58±0.08
FACE/DT	2.63±0.08	2.63±0.07	2.64±0.06	2.69±0.04
FACE/PDT	2.68±0.10	2.70±0.09	2.69±0.08	2.72±0.05
GFG/GG	1.09±0.05	1.06±0.03	1.03±0.04	1.01±0.02
GFG/DT	1.06±0.04	1.03±0.02	1.02±0.03	1.00±0.01
GFG/PDT	1.06±0.05	1.03±0.02	1.02±0.03	1.00±0.01

Table 3 Average hop count and 95% CI with *IG node* for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/IG/GG	15.67±0.91	12.23±0.70	9.72±0.50	8.45±0.39	7.61±0.28
FACE/IG/DT	15.57±0.81	12.04±0.64	9.57±0.44	8.37±0.39	7.59±0.21
FACE/IG/PDT	15.71±0.86	12.15±0.62	9.76±0.46	8.53±0.40	7.73±0.24
GFG/IG/GG	11.58±1.10	7.75±0.80	5.30±0.47	4.46±0.40	3.73±0.20
GFG/IG/DT	11.06±0.99	7.34±0.74	5.11±0.40	4.30±0.36	3.62±0.17
GFG/IG/PDT	11.27±1.05	7.51±0.75	5.16±0.42	4.34±0.37	3.65±0.18
SP	6.63±0.34	5.19±0.24	4.29±0.16	3.82±0.17	3.43±0.09

Table 3 Average hop count and 95% CI with *IG node* for *N*=60 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/IG/GG	6.90±0.24	6.38±0.18	5.73±0.17	5.074±0.099
FACE/IG/DT	6.93±0.17	6.54±0.10	5.95±0.12	5.307±0.068
FACE/IG/PDT	7.05±0.20	6.65±0.13	6.03±0.13	5.355±0.067
GFG/IG/GG	3.26±0.09	2.97±0.05	2.65±0.09	2.256±0.020
GFG/IG/DT	3.22±0.08	2.94±0.04	2.63±0.06	2.254±0.020
GFG/IG/PDT	3.23±0.08	2.94±0.04	2.63±0.07	2.255±0.020
SP	3.12±0.04	2.89±0.04	2.59±0.03	2.247±0.018

Table 4 Average dilation and 95% CI with *IG node* for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/IG/GG	2.36±0.14	2.36±0.13	2.27±0.12	2.21±0.10	2.22±0.08
FACE/IG/DT	2.35±0.12	2.32±0.12	2.23±0.10	2.19±0.10	2.21±0.06
FACE/IG/PDT	2.37±0.13	2.34±0.12	2.28±0.11	2.23±0.10	2.25±0.07
GFG/IG/GG	1.75±0.17	1.49±0.15	1.24±0.11	1.17±0.10	1.09±0.06
GFG/IG/DT	1.67±0.15	1.41±0.14	1.19±0.09	1.13±0.09	1.06±0.05
GFG/IG/PDT	1.70±0.16	1.45±0.14	1.20±0.10	1.14±0.10	1.06±0.05

Table 4 Average dilation and 95% CI with *IG node* for *N*=60 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/IG/GG	2.21±0.08	2.21±0.06	2.21±0.07	2.26±0.04
FACE/IG/DT	2.22±0.05	2.26±0.03	2.30±0.05	2.36±0.03
FACE/IG/PDT	2.26±0.06	2.30±0.04	2.33±0.05	2.38±0.03
GFG/IG/GG	1.04±0.03	1.03±0.02	1.02±0.03	1.00±0.01
GFG/IG/DT	1.03±0.03	1.02±0.01	1.02±0.02	1.00±0.01
GFG/IG/PDT	1.04±0.03	1.02±0.01	1.02±0.03	1.00±0.01

Table 5 Average hop count and 95% CI with *IG node* and shortcut for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/GG/SC	14.24±0.70	10.30±0.63	7.28±0.47	5.83±0.41	4.93±0.27
FACE/DT/SC	15.39±0.83	10.92±0.69	7.77±0.55	6.01±0.43	5.02±0.28
FACE/PDT/SC	15.18±0.86	10.78±0.67	7.64±0.54	5.97±0.42	4.98±0.28
FACE/IG/GG/SC	11.87±0.60	8.64±0.42	6.54±0.31	5.35±0.31	4.61±0.21
FACE/IG/DT/SC	12.51±0.65	9.07±0.45	6.81±0.40	5.52±0.37	4.66±0.20
FACE/IG/PDT/SC	12.32±0.66	8.97±0.43	6.74±0.37	5.48±0.35	4.65±0.21
GFG/GG/SC	10.38±0.81	6.91±0.57	5.01±0.35	4.22±0.34	3.61±0.17
GFG/DT/SC	10.43±0.88	6.83±0.56	5.05±0.36	4.22±0.34	3.58±0.17
GFG/PDT/SC	10.35±0.84	6.89±0.58	5.03±0.35	4.20±0.32	3.59±0.17
GFG/IG/GG/SC	9.19±0.73	6.33±0.47	4.72±0.30	4.06±0.26	3.54±0.12
GFG/IG/DT/SC	9.24±0.74	6.31±0.48	4.72±0.29	4.06±0.26	3.52±0.12
GFG/IG/PDT/SC	9.23±0.75	6.35±0.48	4.72±0.29	4.05±0.25	3.53±0.12
SP	6.63±0.34	5.19±0.24	4.29±0.16	3.82±0.17	3.43±0.09

Table 5 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/GG/SC	4.14±0.14	3.70±0.11	3.10±0.12	2.522±0.050
FACE/DT/SC	4.20±0.16	3.69±0.11	3.06±0.13	2.484±0.067
FACE/PDT/SC	4.18±0.16	3.71±0.12	3.07±0.13	2.481±0.060
FACE/IG/GG/SC	3.97±0.10	3.56±0.09	3.04±0.09	2.485±0.040
FACE/IG/DT/SC	3.99±0.12	3.56±0.08	2.98±0.10	2.438±0.046
FACE/IG/PDT/SC	3.98±0.11	3.57±0.08	3.00±0.10	2.448±0.046
GFG/GG/SC	3.21±0.06	2.95±0.05	2.61±0.05	2.256±0.020
GFG/DT/SC	3.21±0.08	2.94±0.04	2.61±0.05	2.254±0.019
GFG/PDT/SC	3.21±0.08	2.93±0.04	2.61±0.05	2.254±0.019
GFG/IG/GG/SC	3.17±0.05	2.93±0.04	2.61±0.05	2.253±0.020
GFG/IG/DT/SC	3.17±0.05	2.92±0.04	2.61±0.05	2.253±0.019
GFG/IG/PDT/SC	3.17±0.05	2.92±0.04	2.61±0.05	2.253±0.019
SP	3.12±0.04	2.89±0.04	2.59±0.03	2.247±0.018

Table 6 Dilation of average hop and 95% CI with *IG node* and shortcut for N=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE/GG/SC	2.15±0.11	1.98±0.12	1.70±0.11	1.53±0.11	1.44±0.08
FACE/DT/SC	2.32±0.13	2.10±0.13	1.81±0.13	1.57±0.11	1.46±0.08
FACE/PDT/SC	2.29±0.13	2.08±0.13	1.78±0.13	1.56±0.11	1.45±0.08
FACE/IG/GG/SC	1.79±0.09	1.66±0.08	1.52±0.07	1.40±0.08	1.34±0.06
FACE/IG/DT/SC	1.89±0.10	1.75±0.09	1.59±0.09	1.45±0.10	1.36±0.06
FACE/IG/PDT/SC	1.86±0.10	1.73±0.08	1.57±0.09	1.43±0.09	1.36±0.06
GFG/GG/SC	1.57±0.12	1.33±0.11	1.17±0.08	1.10±0.09	1.05±0.05
GFG/DT/SC	1.57±0.13	1.32±0.11	1.18±0.08	1.10±0.09	1.04±0.05
GFG/PDT/SC	1.56±0.13	1.33±0.11	1.17±0.08	1.10±0.08	1.05±0.05
GFG/IG/GG/SC	1.39±0.11	1.22±0.09	1.10±0.07	1.06±0.07	1.03±0.03
GFG/IG/DT/SC	1.39±0.11	1.22±0.09	1.10±0.07	1.06±0.07	1.03±0.03
GFG/IG/PDT/SC	1.39±0.11	1.22±0.09	1.10±0.07	1.06±0.07	1.03±0.03

Table 6 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE/GG/SC	1.33±0.04	1.28±0.04	1.20±0.05	1.12±0.02
FACE/DT/SC	1.35±0.05	1.28±0.04	1.18±0.05	1.11±0.03
FACE/PDT/SC	1.34±0.05	1.28±0.04	1.19±0.05	1.10±0.03
FACE/IG/GG/SC	1.27±0.03	1.23±0.03	1.17±0.03	1.11±0.02
FACE/IG/DT/SC	1.28±0.04	1.23±0.03	1.15±0.04	1.09±0.02
FACE/IG/PDT/SC	1.28±0.04	1.24±0.03	1.16±0.04	1.09±0.02
GFG/GG/SC	1.03±0.02	1.02±0.02	1.01±0.02	1.00±0.01
GFG/DT/SC	1.03±0.03	1.02±0.01	1.01±0.02	1.00±0.01
GFG/PDT/SC	1.03±0.03	1.01±0.01	1.01±0.02	1.00±0.01
GFG/IG/GG/SC	1.02±0.02	1.01±0.01	1.01±0.02	1.00±0.01
GFG/IG/DT/SC	1.02±0.02	1.01±0.01	1.01±0.02	1.00±0.01
GFG/IG/PDT/SC	1.02±0.02	1.01±0.01	1.01±0.02	1.00±0.01

Table 7 Average hop and 95% CI of new *FACE routing* for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE-N/GG	34.59±3.25	23.28±2.33	14.73±1.59	11.39±1.21	9.92±1.01
FACE-N/DT	28.31±2.47	18.36±1.73	11.22±1.04	8.40±0.82	6.93±0.55
FACE-N/PDT	29.95±2.64	19.73±1.95	12.10±1.16	8.96±0.92	7.50±0.63
FACE-N/GG/SC	20.17±1.68	12.81±1.09	8.31±0.69	6.35±0.52	5.34±0.36
FACE-N/DT/SC	19.90±1.66	12.59±1.06	8.22±0.71	6.26±0.52	5.18±0.37
FACE-N/PDT/SC	19.89±1.65	12.63±1.05	8.22±0.70	6.25±0.52	5.17±0.36
FACE-N/IG/GG	23.08±2.13	15.54±1.26	10.71±0.87	8.60±0.63	7.49±0.49
FACE-N/IG/DT	20.91±1.76	13.50±1.09	8.84±0.64	7.00±0.52	5.91±0.36
FACE-N/IG/PDT	21.70±1.87	14.03±1.12	9.34±0.71	7.44±0.58	6.29±0.38
FACE-N/IG/GG/SC	16.72±1.32	10.68±0.74	7.41±0.50	5.85±0.40	4.98±0.29
FACE-N/IG/DT/SC	16.67±1.34	10.56±0.73	7.28±0.48	5.80±0.42	4.86±0.28
FACE-N/IG/PDT/SC	16.68±1.34	10.56±0.72	7.27±0.48	5.79±0.41	4.87±0.28
SP	6.63±0.34	5.19±0.24	4.29±0.16	3.82±0.17	3.43±0.09

Table 7 Average hop and 95% CI of new *FACE routing* for *N*=60 (Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE-N/GG	8.20±0.59	7.03±0.50	5.87±0.38	5.032±0.297
FACE-N/DT	5.74±0.30	5.08±0.28	4.23±0.19	3.629±0.112
FACE-N/PDT	6.16±0.35	5.41±0.33	4.44±0.24	3.753±0.148
FACE-N/GG/SC	4.39±0.17	3.89±0.15	3.21±0.12	2.632±0.069
FACE-N/DT/SC	4.27±0.18	3.78±0.14	3.12±0.12	2.553±0.065
FACE-N/PDT/SC	4.27±0.17	3.79±0.14	3.12±0.12	2.556±0.066
FACE-N/IG/GG	6.44±0.34	5.74±0.23	5.00±0.22	4.421±0.158
FACE-N/IG/DT	5.04±0.18	4.54±0.15	3.90±0.13	3.378±0.068
FACE-N/IG/PDT	5.38±0.23	4.81±0.18	4.07±0.15	3.491±0.085
FACE-N/IG/GG/SC	4.16±0.12	3.75±0.11	3.15±0.09	2.609±0.050
FACE-N/IG/DT/SC	4.10±0.12	3.67±0.10	3.08±0.09	2.546±0.050
FACE-N/IG/PDT/SC	4.09±0.12	3.68±0.10	3.09±0.09	2.556±0.050
SP	3.12±0.04	2.89±0.04	2.59±0.03	2.247±0.018

Table 8 Dilation of Average hop and 95% CI of *new FACE* for *N*=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
FACE-N/GG	5.22±0.49	4.49±0.45	3.43±0.37	2.98±0.32	2.89±0.29
FACE-N/DT	4.27±0.37	3.54±0.33	2.62±0.24	2.20±0.21	2.02±0.16
FACE-N/PDT	4.52±0.40	3.80±0.38	2.82±0.27	2.35±0.24	2.19±0.18
FACE-N/GG/SC	3.04±0.25	2.47±0.21	1.94±0.16	1.66±0.14	1.56±0.10
FACE-N/DT/SC	3.00±0.25	2.43±0.20	1.92±0.17	1.64±0.14	1.51±0.11
FACE-N/PDT/SC	3.00±0.25	2.43±0.20	1.92±0.16	1.64±0.14	1.51±0.10
FACE-N/IG/GG	3.48±0.32	2.99±0.24	2.50±0.2	2.25±0.16	2.18±0.14
FACE-N/IG/DT	3.15±0.27	2.60±0.21	2.06±0.15	1.83±0.14	1.72±0.10
FACE-N/IG/PDT	3.27±0.28	2.70±0.22	2.18±0.17	1.95±0.15	1.83±0.11
FACE-N/IG/GG/SC	2.52±0.20	2.06±0.14	1.72±0.12	1.53±0.10	1.45±0.08
FACE-N/IG/DT/SC	2.51±0.2	2.03±0.14	1.70±0.11	1.52±0.11	1.42±0.08
FACE-N/IG/PDT/SC	2.52±0.20	2.03±0.14	1.69±0.11	1.52±0.11	1.42±0.08

Table 8(Cont')

Name	Degree=9	Degree=10	Degree=12	Degree=15
FACE-N/GG	2.63±0.19	2.43±0.17	2.27±0.15	2.24±0.13
FACE-N/DT	1.84±0.10	1.76±0.1	1.63±0.07	1.62±0.05
FACE-N/PDT	1.97±0.11	1.87±0.11	1.71±0.09	1.67±0.07
FACE-N/GG/SC	1.41±0.05	1.35±0.05	1.24±0.05	1.17±0.03
FACE-N/DT/SC	1.37±0.06	1.31±0.05	1.20±0.05	1.14±0.03
FACE-N/PDT/SC	1.37±0.05	1.31±0.05	1.20±0.05	1.14±0.03
FACE-N/IG/GG	2.06±0.11	1.99±0.08	1.93±0.08	1.97±0.07
FACE-N/IG/DT	1.62±0.06	1.57±0.05	1.51±0.05	1.50±0.03
FACE-N/IG/PDT	1.72±0.07	1.66±0.06	1.57±0.06	1.55±0.04
FACE-N/IG/GG/SC	1.33±0.04	1.30±0.04	1.22±0.03	1.16±0.02
FACE-N/IG/DT/SC	1.31±0.04	1.27±0.03	1.19±0.03	1.13±0.02
FACE-N/IG/PDT/SC	1.31±0.04	1.27±0.03	1.19±0.03	1.14±0.02

Appendix B Experiment for N=60

Table 9: Average size and 95% CI of outer perimeter for N=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
DT	67.9±2.9	52.2±3.2	42.8±2.2	36.6±2.2	30.3±1.5
PDT	72.1±3.1	56.7±3.4	47.6±2.2	41.6±2.2	34.7±1.7
GG	80.9±3.9	66.8±4.2	58.7±3.3	51.7±3.0	44.4±2.3
IG/DT	47.6±2.1	36.7±2.2	29.6±1.4	25.9±1.5	22.3±1.1
IG/PDT	49.4±2.1	38.8±2.3	32.2±1.4	28.6±1.6	24.8±1.3
IG/GG	51.9±2.3	42.2±2.6	35.9±1.7	32.7±2.1	29.4±1.5

Name	Degree=9	Degree=10	Degree=20	Degree=35	Degree=59
DT	26.4±1.4	26.8±1.3	16.1±0.5	12.7±0.5	10.9±0.6
PDT	31.4±1.5	31.1±1.4	21.9±0.5	18.8±0.7	15.1±0.7
GG	41.4±2.5	42.2±2.2	35.9±1.6	36.0±1.7	35.1±1.6
IG/DT	19.7±0.7	18.4±0.8	11.9±0.3	7.9±0.4	0.00±0.00
IG/PDT	22.7±0.8	21.6±1.0	16.2±0.4	10.4±0.6	0.00±0.00
IG/GG	27.3±1.7	26.3±1.2	22.6±0.9	16.5±1.3	0.00±0.00

Table 10: Average size and 95% CI of all perimeters for N=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
DT	4.83±0.09	4.15±0.06	3.78±0.03	3.58±0.03	3.45±0.02
PDT	5.21±0.10	4.45±0.10	3.98±0.04	3.75±0.04	3.57±0.03
GG	7.84±0.44	6.57±0.29	5.77±0.22	5.34±0.18	5.02±0.14
IG/DT	5.90±0.42	4.80±0.21	4.09±0.09	3.76±0.08	3.57±0.05
IG/PDT	6.54±0.68	5.17±0.26	4.35±0.12	3.95±0.10	3.70±0.07
IG/GG	8.27±0.78	6.67±0.53	5.63±0.26	5.23±0.26	4.91±0.20

Name	Degree=9	Degree=10	Degree=20	Degree=35	Degree=59
DT	3.34±0.01	3.32±0.02	3.12±0.00	3.09±0.00	3.07±0.00
PDT	3.45±0.02	3.40±0.02	3.19±0.00	3.15±0.00	3.11±0.00
GG	4.84±0.14	4.83±0.13	4.55±0.09	4.58±0.11	4.59±0.12
IG/DT	3.44±0.03	3.39±0.03	3.15±0.01	3.17±0.01	0.00±0.00
IG/PDT	3.56±0.05	3.52±0.05	3.25±0.01	3.28±0.02	0.00±0.00
IG/GG	4.76±0.18	4.65±0.13	4.42±0.10	4.73±0.24	0.00±0.00

Table 11: Average degree and 95% CI of planar graphs for N=60

Name	Degree=4	Degree=5	Degree=6	Degree=7	Degree=8
DT	3.30±0.04	3.73±0.05	4.10±0.04	4.38±0.06	4.60±0.04
PDT	3.14±0.04	3.53±0.06	3.89±0.04	4.15±0.06	4.39±0.05
GG	2.62±0.05	2.80±0.05	2.98±0.06	3.11±0.06	3.23±0.05
IG/DT	2.93±0.08	3.30±0.09	3.73±0.08	4.07±0.10	4.33±0.08
IG/PDT	2.82±0.09	3.14±0.09	3.53±0.09	3.87±0.10	4.15±0.08
IG/GG	2.56±0.07	2.76±0.07	2.98±0.08	3.12±0.08	3.24±0.08

Name	Degree=9	Degree=10	Degree=20	Degree=35	Degree=59
DT	4.80±0.04	4.85±0.04	5.36±0.01	5.47±0.01	5.53±0.02
PDT	4.60±0.05	4.68±0.05	5.16±0.01	5.27±0.02	5.39±0.02
GG	3.31±0.07	3.31±0.05	3.46±0.05	3.44±0.06	3.44±0.07
IG/DT	4.54±0.07	4.61±0.06	5.13±0.04	4.82±0.09	N/A
IG/PDT	4.34±0.08	4.40±0.08	4.88±0.04	4.56±0.09	N/A
IG/GG	3.31±0.08	3.34±0.06	3.46±0.06	3.15±0.12	N/A

Appendix C Experiment for N=100

Table 12: Average size and 95% CI of outer perimeter for N=100

Name	Degree=5	Degree=6	Degree=7	Degree=8	Degree=9
DT	83.0±3.7	70.1±4.9	56.9±4.4	59.1±6.2	51.6±4.8
PDT	87.0±3.6	75.2±5.8	60.8±5.0	64.1±5.7	57.4±4.3
GG	97.5±3.6	87.2±8.1	73.2±6.3	80.2±7.4	70.3±5.3
IG/DT	58.8±2.7	51.0±3.6	40.9±3.4	43.5±4.0	37.1±2.2
IG/PDT	61.7±2.7	54.9±4.2	42.8±3.3	47.7±3.7	40.7±2.1
IG/GG	67.6±3.3	61.6±5.6	50.3±4.5	55.8±4.8	49.2±2.5

Name	Degree=10	Degree=15	Degree=30	Degree=60	Degree=99
DT	49.1±3.2	31.4±2.3	19.8±1.1	14.8±1.0	12.0±1.6
PDT	53.6±3.2	36.9±3.2	24.9±1.4	22.0±1.4	17.3±0.8
GG	67.4±3.9	50.6±4.6	45.2±3.7	42.5±4.1	47.9±3.3
IG/DT	35.9±1.8	24.5±1.8	15.6±0.9	9.2±0.8	0.0±0.0
IG/PDT	40.0±1.9	29.1±2.5	20.2±1.5	12.5±1.1	0.0±0.0
IG/GG	48.1±2.8	37.9±3.4	31.5±3.3	22.7±2.5	0.0±0.0

Table 13: Average size and 95% CI of all perimeters for N=100

Name	Degree=5	Degree=6	Degree=7	Degree=8	Degree=9
DT	4.05±0.05	3.80±0.04	3.59±0.04	3.50±0.05	3.42±0.04
PDT	4.20±0.07	3.95±0.05	3.69±0.05	3.58±0.05	3.50±0.04
GG	5.43±0.22	5.45±0.26	4.97±0.20	4.91±0.16	4.59±0.11
IG/DT	4.03±0.09	3.85±0.07	3.65±0.07	3.51±0.06	3.42±0.04
IG/PDT	4.18±0.11	4.03±0.10	3.76±0.08	3.61±0.05	3.50±0.04
IG/GG	5.20±0.28	5.18±0.23	4.77±0.17	4.74±0.13	4.47±0.17

Name	Degree=10	Degree=15	Degree=30	Degree=60	Degree=99
DT	3.37±0.02	3.19±0.02	3.09±0.00	3.06±0.00	3.04±0.0
PDT	3.42±0.03	3.24±0.03	3.12±0.00	3.10±0.00	3.07±0.0
GG	4.58±0.16	4.42±0.11	4.31±0.09	4.33±0.11	4.47±0.1
IG/DT	3.36±0.03	3.18±0.02	3.10±0.00	3.11±0.01	0.0±0.0
IG/PDT	3.43±0.04	3.24±0.03	3.15±0.01	3.18±0.02	0.0±0.0
IG/GG	4.43±0.16	4.34±0.14	4.26±0.08	4.61±0.18	0.0±0.0

Table 14: Average degree and 95% CI of planar graphs for N=100

Name	Degree=5	Degree=6	Degree=7	Degree=8	Degree=9
DT	3.87±0.05	4.13±0.05	4.41±0.06	4.58±0.09	4.71±0.08
PDT	3.74±0.05	3.97±0.05	4.28±0.07	4.44±0.08	4.57±0.07
GG	3.11±0.07	3.11±0.08	3.29±0.08	3.31±0.07	3.47±0.06
IG/DT	3.87±0.09	4.05±0.08	4.30±0.10	4.51±0.10	4.68±0.08
IG/PDT	3.74±0.09	3.87±0.09	4.16±0.11	4.36±0.08	4.54±0.07
IG/GG	3.18±0.10	3.18±0.08	3.36±0.08	3.37±0.07	3.54±0.11

Name	Degree=10	Degree=15	Degree=30	Degree=60	Degree=99
DT	4.82±0.05	5.25±0.06	5.54±0.02	5.64±0.02	5.70±0.03
PDT	4.71±0.06	5.12±0.07	5.44±0.02	5.49±0.03	5.59±0.01
GG	3.49±0.09	3.58±0.07	3.66±0.06	3.64±0.08	3.55±0.08
IG/DT	4.81±0.06	5.23±0.07	5.45±0.02	5.23±0.08	N/A
IG/PDT	4.67±0.09	5.09±0.08	5.31±0.02	5.02±0.08	N/A
IG/GG	3.56±0.10	3.62±0.09	3.66±0.06	3.32±0.13	N/A

Appendix D Experiment for N=200

Table 15: Average size and 95% CI of outer perimeter for N=200

Name	Degree=7	Degree=8	Degree=9	Degree=10	Degree=12
DT	101.2±6.8	92.3±9.3	80.9±6.2	77.5±6.3	65.0±5.6
PDT	106.6±8.4	98.3±9.2	87.4±6.1	84.5±7.5	73.4±5.5
GG	119.1±11.0	117.7±11.4	106.4±8.5	105.5±7.8	96.9±8.9
IG/DT	75.5±4.58	68.1±6.7	59.2±4.8	57.8±3.5	50.4±4.3
IG/PDT	79.3±5.83	72.9±6.8	65.0±4.7	63.8±5.0	57.5±4.4
IG/GG	89.7±7.68	82.3±8.4	74.7±5.8	75.9±6.5	71.0±6.1

Name	Degree=15	Degree=25	Degree=50	Degree=100	Degree=199
DT	55.9±4.9	33.3±1.5	21.9±1.4	18.3±1.0	14.9±1.6
PDT	64.4±5.5	41.8±0.8	32.9±1.4	30.2±1.6	23.7±1.0
GG	83.2±6.8	66.6±4.2	64.6±4.6	66.3±4.4	69.5±4.5
IG/DT	42.7±3.4	26.2±1.0	18.0±0.5	11.2±0.3	0.00±0.0
IG/PDT	48.2±3.8	33.1±0.5	26.6±0.9	15.8±0.9	0.00±0.0
IG/GG	59.6±4.7	51.0±4.3	47.0±3.7	36.6±2.8	0.00±0.0

Table 16: Average size and 95% CI of all perimeters for N=200

Name	Degree=7	Degree=8	Degree=9	Degree=10	Degree=12
DT	3.52±0.01	3.47±0.02	3.38±0.02	3.33±0.02	3.25±0.02
PDT	3.58±0.02	3.54±0.02	3.44±0.02	3.38±0.03	3.30±0.02
GG	4.46±0.08	4.73±0.09	4.56±0.08	4.54±0.16	4.44±0.11
IG/DT	3.49±0.03	3.48±0.02	3.39±0.02	3.32±0.02	3.26±0.02
IG/PDT	3.54±0.03	3.55±0.03	3.46±0.03	3.38±0.04	3.31±0.02
IG/GG	4.37±0.06	4.59±0.09	4.43±0.06	4.40±0.15	4.31±0.12

Name	Degree=15	Degree=25	Degree=50	Degree=100	Degree=199
DT	3.19±0.02	3.08±0.00	3.05±0.00	3.04±0.00	3.03±0.00
PDT	3.23±0.02	3.11±0.00	3.08±0.00	3.07±0.00	3.05±0.00
GG	4.39±0.07	4.25±0.07	4.18±0.05	4.23±0.06	4.26±0.07
IG/DT	3.19±0.02	3.08±0.00	3.06±0.00	3.05±0.00	0.00±0.00
IG/PDT	3.22±0.03	3.11±0.00	3.09±0.00	3.08±0.01	0.00±0.00
IG/GG	4.25±0.10	4.13±0.08	4.16±0.05	4.27±0.11	0.00±0.00

Table 17: Average degree and 95% CI of planar graphs for N=200

Name	Degree=7	Degree=8	Degree=9	Degree=10	Degree=12
DT	4.57±0.03	4.67±0.04	4.84±0.04	4.95±0.06	5.13±0.05
PDT	4.47±0.03	4.55±0.03	4.71±0.04	4.83±0.07	5.02±0.05
GG	3.59±0.05	3.43±0.05	3.52±0.05	3.55±0.09	3.60±0.07
IG/DT	4.62±0.05	4.64±0.04	4.81±0.05	4.95±0.05	5.09±0.05
IG/PDT	4.53±0.05	4.51±0.05	4.67±0.06	4.82±0.08	4.97±0.06
IG/GG	3.64±0.04	3.50±0.05	3.59±0.04	3.63±0.09	3.69±0.09

Name	Degree=15	Degree=25	Degree=50	Degree=100	Degree=199
DT	5.30±0.06	5.62±0.02	5.74±0.01	5.78±0.01	5.81±0.01
PDT	5.19±0.07	5.53±0.01	5.64±0.01	5.66±0.01	5.73±0.01
GG	3.64±0.05	3.74±0.05	3.79±0.04	3.75±0.05	3.73±0.06
IG/DT	5.29±0.07	5.61±0.01	5.68±0.00	5.65±0.02	N/A
IG/PDT	5.19±0.07	5.51±0.01	5.56±0.01	5.54±0.04	N/A
IG/GG	3.73±0.08	3.83±0.07	3.79±0.04	3.67±0.08	N/A

Appendix E Experiment for N=500

Table 18: Average size and 95% CI of outer perimeter for N=500

Name	Degree=9	Degree=10	Degree=12	Degree=15	Degree=20
DT	158.0±20.8	157.6±9.4	143.6±13.9	106.0±7.5	88.6±2.3
PDT	169.0±21.8	171.3±10.6	155.3±12.8	116.0±6.9	99.0±0.9
GG	193.0±31.0	203.3±7.6	185.3±9.8	146.0±6.4	131.3±6.5
IG/DT	114.3±10.1	117.0±12.9	111.6±11.1	83.3±1.9	67.6±1.9
IG/PDT	121.6±12.3	129.6±14.1	120.6±13.6	92.6±3.7	77.0±0.9
IG/GG	137.6±20.1	150.0±8.8	139.6±10.5	112.3±4.6	94.0±6.6

Name	Degree=40	Degree=75	Degree=150	Degree=250	Degree=499
DT	52.0±7.3	36.3±2.6	23.3±2.3	24.0±1.8	21.3±1.9
PDT	66.6±8.0	50.3±5.0	42.0±0.0	36.6±1.4	30.6±3.7
GG	109.3±8.3	101.0±11.7	101.6±7.9	98.6±7.4	106.3±9.1
IG/DT	40.6±4.1	28.3±1.4	17.6±0.5	11.6±1.4	0.00±0.00
IG/PDT	55.3±1.9	38.6±4.1	29.3±1.4	18.0±0.9	0.00±0.00
IG/GG	87.3±8.7	77.3±5.5	72.6±6.1	55.0±8.9	0.00±0.00

Table 19: Average size of all perimeters and 95% CI for N=500

Name	Degree=9	Degree=10	Degree=12	Degree=15	Degree=20
DT	3.34±0.01	3.29±0.01	3.24±0.00	3.13±0.02	3.10±0.00
PDT	3.40±0.00	3.33±0.01	3.28±0.00	3.15±0.01	3.12±0.00
GG	4.39±0.00	4.34±0.08	4.23±0.09	4.16±0.04	4.08±0.01
IG/DT	3.32±0.03	3.27±0.01	3.22±0.01	3.14±0.02	3.09±0.00
IG/PDT	3.38±0.02	3.31±0.02	3.26±0.00	3.17±0.02	3.11±0.00
IG/GG	4.29±0.03	4.24±0.07	4.12±0.09	4.06±0.01	3.99±0.03

Name	Degree=40	Degree=75	Degree=150	Degree=250	Degree=499
DT	3.05±0.00	3.03±0.00	3.02±0.00	3.02±0.00	3.01±0.00
PDT	3.07±0.00	3.05±0.00	3.04±0.00	3.03±0.00	3.02±0.00
GG	4.10±0.04	4.12±0.07	4.11±0.01	4.09±0.06	4.13±0.03
IG/DT	3.05±0.00	3.03±0.00	3.02±0.00	3.02±0.00	0.00±0.00
IG/PDT	3.07±0.00	3.04±0.00	3.03±0.00	3.04±0.00	0.00±0.00
IG/GG	4.08±0.03	4.07±0.07	4.13±0.04	4.18±0.15	0.00±0.00

Table 20: Average degree and 95% CI for N=500

Name	Degree=9	Degree=10	Degree=12	Degree=15	Degree=20
DT	4.95±0.03	5.08±0.03	5.20±0.01	5.43±0.06	5.59±0.01
PDT	4.82±0.01	4.97±0.03	5.10±0.00	5.37±0.05	5.53±0.00
GG	3.65±0.00	3.68±0.05	3.78±0.07	3.83±0.03	3.89±0.01
IG/DT	4.99±0.06	5.11±0.03	5.23±0.03	5.45±0.07	5.63±0.01
IG/PDT	4.86±0.04	5.01±0.04	5.14±0.01	5.39±0.07	5.56±0.01
IG/GG	3.72±0.02	3.76±0.06	3.86±0.08	3.92±0.00	3.99±0.03

Name	Degree=40	Degree=75	Degree=150	Degree=250	Degree=499
DT	5.76±0.02	5.84±0.01	5.89±0.00	5.89±0.00	5.90±0.00
PDT	5.70±0.02	5.78±0.02	5.81±0.00	5.84±0.00	5.86±0.01
GG	3.88±0.03	3.86±0.06	3.87±0.01	3.89±0.06	3.86±0.02
IG/DT	5.77±0.01	5.85±0.00	5.88±0.00	5.84±0.00	N/A
IG/PDT	5.70±0.00	5.80±0.01	5.81±0.01	5.77±0.02	N/A
IG/GG	3.89±0.02	3.91±0.06	3.85±0.04	3.79±0.12	N/A

Appendix F Experiment for N=1000

Table 21: Size of outer perimeter for N=1000

Name	Degree=12	Degree=13	Degree=15	Degree=20	Degree=30
DT	215.0±13.2	205.0±11.9	181.4±14.2	145.6±8.2	91.6±5.5
PDT	230.6±12.8	221.2±11.8	198.8±16.5	163.0±8.4	108.4±6.4
GG	273.4±14.7	260.2±7.4	241.0±24.0	218.0±12.1	160.2±18.3
IG/DT	162.6±10.9	151.2±9.2	139.0±9.2	113.6±4.1	73.2±1.9
IG/PDT	174.6±9.9	163.4±9.5	154.6±12.2	129.6±3.9	90.4±5.5
IG/GG	200.2±12.6	188.0±6.9	183.0±16.7	171.6±4.3	126.4±13.5

Name	Degree=60	Degree=120	Degree=250	Degree=500	Degree=999
DT	58.8±2.3	39.8±3.4	25.6±1.7	27.0±2.1	21.2±1.9
PDT	75.4±2.8	65.2±2.1	56.2±3.1	52.6±3.0	38.4±3.1
GG	128.8±6.0	144.2±5.4	155.6±8.6	160.0±5.9	165.0±10.0
IG/DT	48.4±2.5	33.4±2.8	21.6±0.8	12.4±0.8	0.00±0.0
IG/PDT	64.6±1.4	53.4±1.8	37.2±1.4	19.0±0.9	0.00±0.0
IG/GG	104.8±7.0	113.4±4.0	96.6±3.7	58.4±7.2	0.00±0.0

Table 22: Average size of inner perimeters for N=1000

Name	Degree=12	Degree=13	Degree=15	Degree=20	Degree=30
DT	3.19±0.01	3.17±0.01	3.13±0.00	3.09±0.00	3.05±0.00
PDT	3.22±0.00	3.19±0.01	3.15±0.01	3.11±0.00	3.06±0.00
GG	4.05±0.05	4.00±0.02	4.01±0.03	3.99±0.05	4.02±0.04
IG/DT	3.18±0.00	3.15±0.01	3.12±0.00	3.08±0.00	3.04±0.00
IG/PDT	3.20±0.00	3.17±0.01	3.14±0.00	3.09±0.00	3.06±0.00
IG/GG	3.97±0.04	3.93±0.02	3.95±0.01	3.93±0.05	3.97±0.04

Name	Degree=60	Degree=120	Degree=250	Degree=500	Degree=999
DT	3.02±0.00	3.01±0.00	3.01±0.00	3.01±0.00	3.00±0.0
PDT	3.03±0.00	3.03±0.00	3.02±0.00	3.02±0.00	3.01±0.0
GG	4.02±0.02	4.04±0.03	4.06±0.01	4.09±0.04	4.07±0.0
IG/DT	3.02±0.00	3.01±0.00	3.01±0.00	3.01±0.00	0.00±0.0
IG/PDT	3.03±0.00	3.02±0.00	3.02±0.00	3.02±0.00	0.00±0.0
IG/GG	3.97±0.02	4.02±0.04	4.02±0.03	4.19±0.06	0.00±0.0

Table 23: Average degree for N=1000

Name	Degree=12	Degree=13	Degree=15	Degree=20	Degree=30
DT	5.32±0.02	5.40±0.03	5.50±0.02	5.64±0.02	5.78±0.00
PDT	5.25±0.02	5.33±0.02	5.44±0.03	5.59±0.02	5.74±0.01
GG	3.94±0.05	3.98±0.02	3.97±0.03	3.99±0.05	3.97±0.04
IG/DT	5.37±0.02	5.46±0.03	5.54±0.01	5.67±0.01	5.80±0.00
IG/PDT	5.29±0.02	5.38±0.03	5.48±0.02	5.62±0.02	5.75±0.00
IG/GG	4.01±0.04	4.05±0.02	4.03±0.01	4.06±0.05	4.02±0.04

Name	Degree=60	Degree=120	Degree=250	Degree=500	Degree=999
DT	5.87±0.00	5.91±0.00	5.94±0.00	5.94±0.00	5.95±0.00
PDT	5.84±0.00	5.86±0.00	5.88±0.00	5.88±0.00	5.91±0.00
GG	3.97±0.02	3.94±0.03	3.92±0.01	3.90±0.03	3.92±0.03
IG/DT	5.88±0.00	5.91±0.00	5.93±0.00	5.90±0.00	N/A
IG/PDT	5.84±0.00	5.87±0.00	5.88±0.00	5.87±0.00	N/A
IG/GG	4.02±0.02	3.96±0.04	3.96±0.03	3.80±0.05	N/A

Appendix G Experiment for N=1600

Table 24: Size of outer perimeter

Name	Degree=12	Degree=15	Degree=20	Degree=25	Degree=50
DT	289.4±12.8	234.4±24.6	167.8±10.9	146.2±6.8	89.4±6.2
PDT	308.8±13.9	252.0±27.4	188.2±9.1	168.6±9.4	110.4±8.0
GG	357.8±19.0	303.2±35.4	235.4±17.9	223.8±15.2	168.8±7.7
IG/DT	217.0±12.3	177.0±16.5	132.6±6.1	117.0±5.7	73.6±4.0
IG/PDT	234.8±15.2	191.4±18.3	149.8±6.1	134.4±7.9	95.0±4.6
IG/GG	270.8±18.7	225.6±22.8	184.6±10.0	175.8±13.7	144.8±5.3

Name	Degree=100	Degree=200	Degree=400	Degree=800	Degree=1599
DT	59.0±3.6	38.6±1.6	31.0±1.9	30.0±1.9	26.4±0.8
PDT	87.2±5.7	75.6±2.5	67.4±2.2	61.8±2.3	46.0±1.8
GG	176.2±9.6	187.0±4.9	195.6±14.1	196.0±3.5	203.4±10.4
IG/DT	49.8±1.2	30.2±0.6	22.0±1.5	12.8±0.6	0.0±0.0
IG/PDT	76.4±4.2	55.8±3.4	41.8±2.0	21.4±1.8	0.0±0.0
IG/GG	140.4±6.3	140.0±8.9	117.8±7.9	68.8±5.8	0.0±0.0

Table 25: Average size of inner perimeters

Name	Degree=12	Degree=15	Degree=20	Degree=25	Degree=50
DT	3.16±0.00	3.11±0.00	3.08±0.00	3.05±0.00	3.02±0.00
PDT	3.18±0.01	3.12±0.01	3.09±0.00	3.06±0.00	3.03±0.00
GG	3.88±0.02	3.80±0.02	3.83±0.03	3.84±0.03	3.90±0.03
IG/DT	3.14±0.00	3.09±0.00	3.07±0.00	3.05±0.00	3.02±0.00
IG/PDT	3.17±0.00	3.11±0.00	3.08±0.00	3.06±0.00	3.03±0.00
IG/GG	3.83±0.01	3.75±0.01	3.79±0.03	3.80±0.02	3.88±0.03

Name	Degree=100	Degree=200	Degree=400	Degree=800	Degree=1599
DT	3.01±0.00	3.01±0.00	3.00±0.00	3.00±0.00	3.00±0.00
PDT	3.02±0.00	3.02±0.00	3.02±0.00	3.01±0.00	3.01±0.00
GG	3.95±0.02	4.02±0.01	4.04±0.02	4.04±0.03	4.04±0.01
IG/DT	3.01±0.00	3.01±0.00	3.00±0.00	3.01±0.00	0.00±0.00
IG/PDT	3.02±0.00	3.01±0.00	3.01±0.00	3.02±0.00	0.00±0.00
IG/GG	3.93±0.01	4.00±0.01	4.03±0.02	4.10±0.08	0.00±0.00

Table 26: Average degree

Name	Degree=12	Degree=15	Degree=20	Degree=25	Degree=50
DT	5.42±0.02	5.58±0.02	5.69±0.02	5.77±0.00	5.87±0.00
PDT	5.35±0.02	5.53±0.03	5.65±0.02	5.73±0.01	5.84±0.01
GG	4.11±0.02	4.21±0.02	4.18±0.03	4.16±0.03	4.09±0.03
IG/DT	5.48±0.02	5.63±0.01	5.72±0.01	5.79±0.01	5.89±0.00
IG/PDT	5.41±0.02	5.58±0.02	5.68±0.01	5.76±0.01	5.86±0.00
IG/GG	4.17±0.02	4.26±0.02	4.22±0.04	4.20±0.02	4.12±0.03

Name	Degree=100	Degree=200	Degree=400	Degree=800	Degree=1599
DT	5.92±0.00	5.94±0.00	5.95±0.00	5.95±0.00	5.96±0.00
PDT	5.88±0.00	5.90±0.00	5.91±0.00	5.91±0.00	5.93±0.00
GG	4.04±0.02	3.96±0.01	3.95±0.02	3.95±0.02	3.95±0.01
IG/DT	5.92±0.00	5.95±0.00	5.95±0.00	5.93±0.00	N/A
IG/PDT	5.89±0.00	5.91±0.00	5.91±0.00	5.89±0.00	N/A
IG/GG	4.06±0.02	3.98±0.01	3.95±0.02	3.88±0.06	N/A