RANDOMIZED ROUTING ALGORITHMS IN MOBILE AD HOC NETWORKS

ISRAAT TANZEENA HAQUE

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2004

Your file Votre référence
ISBN: 0-612-94741-6
Our file Notre référence
ISBN: 0-612-94741-6

Abstract

Randomized Routing Algorithms in Mobile Ad Hoc Networks

Israat Tanzeena Haque

We consider the problem of finding a path from a source to a destination node in a set of mobile wireless nodes. Many solutions to this problem proposed in the literature fall into the *position-based* routing paradigm, where in each step, the decision of which node to go to next is based only on the position or geographic coordinates of the current node c, its neighboring nodes $N(c)$, and the destination node d. We propose several new randomized position-based algorithms for routing in mobile ad hoc networks. Our algorithms combine the greedy heuristic of minimizing the distance remaining to the destination and the directional heuristic of staying close to the direction of the destination with the use of randomization to retain some flexibility in the chosen routes. We classify our randomized algorithms based on the strategy they use to define a subset of neighboring nodes as the candidate nodes. The *sector-based* algorithms select the candidate nodes from a specified sector, whereas the *AB (above-below)* algorithms choose two candidate nodes, one from above and the other from below the line between the current node and the destination. On convex subdivisions, a sub-class of *AB* algorithms can be shown to deliver packets to their destinations with probability 1. Our experiments on unit disk graphs, and their associated Yao graphs, Gabriel graphs, and Planarized Local Delaunay Triangulations, show that the delivery rates of all the randomized algorithms we study are significantly better than the deterministic greedy and directional routing algorithms. For some of the algorithms we propose, this improvement comes at the price of only a small deterioration in the stretch factor of the route. Thus, some of our algorithms obtain a good balance between the delivery rate and the stretch factor.

Acknowledgments

I would like to thank some people without whom it would not have been possible for me to finish this thesis. First and foremost, I would like to express my thankfulness to my supervisors Dr. Lata Narayanan and Dr. Thomas Fevens for their continuous concern about my work. Their valuable suggestions helped me to learn how to analyze a problem, and then solve it step by step using both theoretical and practical analysis. Secondly, I would like to thank my friends for their support, especially all the group discussions with me. Anup Patnaik, Sabeel Ansari and Dongwook Cho helped me a lot to clear my doubts. Finally, I want to thank Dr. Jit Bose and Dr. Pat Morin of Carleton University and Jie Gao of Stanford University for sharing their information.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last couple of decades, mobile computing has become very popular because of the continued miniaturization of mobile computing devices and the extraordinary rise of processing power available in mobile laptop computers, which in turn has lead to more and better computer-based applications in the hands of a growing number of mobile computer/device users. At the same time, the markets for wireless telephones and communication devices have expanded rapidly. As a result, these users have started to expect having networking applications in situations where the Internet itself is not available or not efficient to use. For instance, a rescue team might be working in a remote flooded area, and the members might want to communicate among themselves without the mediation of routing across the global Internet [Per01]. However, it is not possible to provide such communication by using Internet protocols, where dedicated static routers with plenty of resources provide the routing services. A solution can be found by using an infrastructureless wireless network called a *Mobile Ad hoc Network (MANET)*. Notably, one of the original motivations for ad hoc networks was found in military applications [Raj02].

A MANET is a collection of autonomous mobile devices that can communicate with each other without having any fixed infrastructure. Each node in the network has an omnidirectional antenna and can communicate using wireless broadcasts with all nodes within its transmission range. Thus a MANET can be represented by a *unit disk graph (UDG)*, where two nodes are connected if and only if their Euclidean distance is at most the transmission range [BFNO01]. Since nodes may not communicate directly with all other nodes because of the limited transmission range, multi-hop communication is required in the network. The nature of MANETs gives rise to issues such as dynamic topology changes,

absence of infrastructure, autonomous heterogeneous nodes, and resource constraints, that contribute to making the problem of routing in these networks a tremendous challenge.

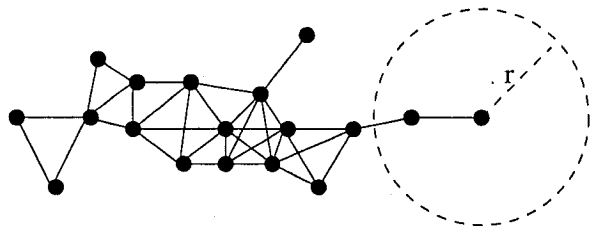

Figure 1.1: A unit disk graph representation of MANET.

1.1 MANET Evolution and Applications

MANETs were first implemented as Packet Radio Networks (PRNET) in 1972, sponsored by the Department of Defense (DoD), and evolved into the Survivable Adaptive Radio Networks (SURAN) program in the early 1980s [FL01]. The goals of these programs were to provide packet-switched networking to mobile battlefield elements in an infrastructureless, hostile environment. The concept of commercial ad hoc networking had arrived in the early 1990s because of the popularity of notebook computers, availability of open-source software and viable communications equipment based on radio frequency and infrared. At around the same time, DoD again started their program Global Mobile (GloMo) Information Systems [RR02].

Several of commercial standards were developed in the '90s due to the growing interest in ad hoc networking. The MANET working group was initiated by the Internet Engineering Task Force (IETF) and its primary purpose was to standardize routing protocols for ad hoc networks. Other than that, a medium access protocol based on collision avoidance and one that tolerated hidden terminals, was standardized by the IEEE 802.11 subcommittee. HIPERLAN and Bluetooth were some other standards that addressed and benefited ad hoc networking [RR02]. Most of the existing ad hoc networks outside the military domain have been developed in the academic environment; however, recently commercially oriented solutions have started to appear (for example, MeshNetworks[1] and SPANworks[2]) [CCL03].

[1]http://www.meshnetworks.com
[2]http://www.spanworks.com

The advantages of using ad hoc networks over traditional wireless networks are ease and speed of deployment, and decreased dependence on a fixed infrastructure. MANETs are suited for use in situations where infrastructure is either not available, not trusted, or should not be relied on in times of emergency. A few examples include: military soldiers in the battlefield, sensors scattered throughout a city for detection of biologically harmful elements, an infrastructure-less network of notebook computers in a conference or campus setting, rare animal tracking, space exploration, undersea operations, and temporary offices such as campaign headquarters [RR02].

1.2 Ad Hoc Networking Issues

Ad hoc networks inherit the standard problems of wireless communication and networking such as vulnerable and unreliable wireless medium, unprotected channels, and hidden and exposed terminal phenomena. Also, the absence of infrastructure, dynamically changing network topology, and the multihop nature of MANETs introduces a number of characteristics, complexities, and design constraints that are specific to ad hoc networking [CCL03]. We will now briefly describe some of these characteristics and issues. The detailed description can be found in [CCL03].

Lack of infrastructure: The main advantage of using a MANET is that it does not depend on any fixed infrastructure or centralized administration. Hence, each node needs to operate in a distributed peer-to-peer mode, acts as an independent router and generates independent data. These characteristics in turn make network management more complicated in MANETs.

Multi-hop routing: Mobile devices have limited radio range and there is no dedicated router in such an environment. Nodes send/receive their own messages as well as forward others' packets thus acting as routers. Hence, multi-hop routing is obviously necessary in MANETs.

Dynamically changing network topologies: In mobile ad hoc networks, nodes move frequently and unpredictably, which causes the network topology to change dynamically. The resulting effect would be route changes, frequent network partitions, and possibly packet losses.

Variation in link and node capabilities: In a MANET, each node might be equipped with one or more radio interfaces with different configurations such as different transmission

power and frequency bands [CL85, CL87]. This heterogeneity in nodes' radio capabilities can result in asymmetric links. In addition, each mobile node might have different processing capabilities because of varying software/hardware configurations. Efficient network protocols and algorithms for such a heterogeneous network should be able to adapt to the changing network conditions (power and channel conditions, traffic load/distribution variations, and congestion)

Energy constrained operation: Since mobile nodes have limited power supply and processing power, the services and applications that can be supported by these nodes are limited. However, each node needs to forward others' packets as well as generate its own messages, which means that additional power is needed for the first task. Currently, this is an important issue in MANET research.

Network scalability: Most of the existing network management algorithms were designed for fixed or relatively small wireless networks. However, there are some situations where we need to consider a large number of wireless nodes, such as sensor and tactical networks [FL01]. Scalability is very important in the successful designing of such networks, where nodes have limited resources. Network designer should consider issues such as addressing, routing, location management, configuration management, interoperability, security, and high-capacity wireless technologies.

Limited physical security: Mobile devices are naturally more vulnerable to security breaches than are static devices. Typical examples include: devices or data can be stolen and information can be taken or modified by the intruder during routing. Hence, design of a network for such devices should consider eavesdropping, spoofing and denial-of-service attacks.

1.3 Routing in MANETs

In ad hoc wireless networks two hosts can communicate if they are within wireless transmission range of each other without using any routing protocol. However, the power needed to obtain a fully connected network topology may be impractical, wasteful, too vulnerable to detection, or even illegal. Hence, combined with the lack of infrastructure routers, the restricted range of wireless transmission indicates the need for multihop routing [Per01]. Moreover, the mobility of nodes, low bandwidth, and low power poses special challenges for routing. The volatility of network information increases because of the topological changes which are caused by node mobility. Also, wireless networks work with lower

bandwidth than wired networks, and hence, information collection for the formation of a routing table is expensive. In addition, the limitation of power leads users to disconnect mobile units frequently in order to reduce power consumption. As a result, traditional routing protocols in wired networks, that generally use either link-state or distance vector, are not suitable for ad hoc wireless networks.

In the last few years, a plethora of routing protocols for MANETs has been proposed in the literature. As yet there is no consensus and no standards have been adopted. The proposed protocols can be divided into two main categories: proactive and reactive protocols. Proactive or table-driven protocols [PB94] are based on Internet distance-vector and link-state protocols, and maintain consistent and updated routing information about the entire network by exchanging information periodically. Randomized versions of some of these proactive protocols have been proposed, such as R-DSDV [CD02]. Reactive or on-demand routing protocols [PR99, JMHJ02] discover routes only when data needs to be sent or the topology is changed. Reactive protocols typically use less bandwidth in terms of control packets to discover topology information, but even so, packets to discover new routes must sometimes be flooded through the network, which consumes a huge amount of bandwidth [CCL03].

One way of limiting flooding is by using information about the position of nodes in the network. In position-based routing protocols, a node forwards packets based on the location (coordinates in the plane) of itself, its neighbors, and the destination [GSB03]. For example, the position of the nodes can be obtained using GPS. There are numerous ways of using position information in making routing decisions.

For example, *Greedy* (distance) routing moves the packet to a neighboring node to minimize the distance towards the destination and *Compass* (directional) routing minimizes the angle formed between the current node, the next node, and the destination node. Both algorithms forward the packet in every step to exactly one of its neighbors, which is chosen according to either of the above mentioned heuristics. However, both of these algorithms are known to fail to deliver the packet in certain situations which are explained in detail in Chapter 2. Also, position information can be used to extract a planar subgraph of the network such that routing can be performed on the perimeter of faces of this subgraph as in [BMSJ99] and [KK00]. The advantage of this latter approach is that delivery of packets can always be guaranteed. Two possible performance measures for a routing algorithm are the delivery rate and lengths of routes. In MANETs we are interested in the ratio of the

number of hops taken by the routing algorithm to the minimum number of hops needed to reach to the destination. This ratio is referred to as *stretch factor.*

The underlying network topology also plays an important role in routing. Some desirable properties of a network topology are distributed construction, connectivity, sparseness, bounded node degree, and small spanning ratio. Most of the position-based routing algorithms use a *UDG* as the network topology. However, some position-based algorithms need a planar subgraph of a *UDG* to ensure guaranteed packet delivery. *Relative Neighborhood Graph (RNG)* [Tou80] and *Gabriel Graph (GG)* [GS69] are two such graphs. In [BM99] Bose and Morin proved that *Greedy*, *Compass*, and *Random Compass* algorithms guarantee delivery in a *Delaunay Triangulation (DT)*, which is another planar graph. However, a *DT* is very expensive to construct locally, and does not respect wireless transmission ranges. So it is not suitable for MANETs. Li *et al.* in [LCW02] propose a novel distributed algorithm to construct a DT locally. The resulting graph is called a *Planarized Localized DT (PLDel)*. Recently the Yao graph, which is not a planar graph, has become very popular for topology control in MANETs of directional antennas [Li03b].

1.4 Contribution of Thesis

In this thesis, we propose several variations of the greedy and directional heuristics. To improve on the delivery rate of these algorithms, we use randomization. Our heuristics combine in various ways the two goals of covering as much distance as possible to the destination (as in greedy routing) and staying as close as possible to the direction of the destination (as in directional routing), while using randomization to allow flexibility with respect to the actual path followed. The randomization increases the delivery rate by introducing flexibility in choosing the next route node; however, it may not guarantee an optimal stretch factor if the next node is chosen uniformly at random out of the current node's neighbors. In this thesis we introduce different methods of assigning weights to the neighboring nodes to keep the stretch factor as low as possible and at the same time ensure higher a delivery rate.

We evaluate all of our algorithms in terms of delivery rate and stretch factor on the *GG*, *PLDel*, and Yao subgraphs of the *UDG*. For purposes of comparison, we also study the performance of *Greedy* and *Compass* routing algorithms.

1.5 Organization of Thesis

In Chapter 2 we discuss existing routing algorithms for mobile ad hoc networks with the emphasis on position-based routing. *Greedy* and *Compass* routings are known to fail to deliver the packet in certain cases. We discuss those situations with examples. Also, we discuss the routing algorithms that guarantee packet delivery. These algorithms require planar graphs as the network topology. We review those planar graphs along with the Yao graph as the network topology for MANETs.

Chapter 3 gives the problem definition and network model. Then, we classify our proposed randomized algorithms based on the way of selecting the eligible candidate nodes. We define four classes of algorithms: *Best two neighbors*, *Sector*, *AB (Above-Below)*, and a class combining the sector and *AB* ideas. Next we define all of our randomized algorithms. Finally, we present the attractive properties that our randomized algorithms possess.

Chapter 4 first describes our own simulator written in using C++. We present the simulation environment with details about simulation parameters. The detailed discussion of the simulation results of all the algorithms, on different graphs, are given next.

In Chapter 5 we briefly discuss our contributions to the research on MANETs. We have discovered several interesting and challenging open problems during the research for this thesis, which are also described in the end of this final chapter.

Chapter 2

Routing in Ad Hoc Networks

In this chapter, we discuss existing routing algorithms for mobile ad hoc networks with an emphasis on position-based routing. *Greedy* and *directional* routing fail to deliver packets in certain scenarios. We discuss those cases with examples. Also, we discuss the routing algorithms that do guarantee packet delivery. These algorithms require the network topology to be a planar graph. All the commonly used topologies for position-based routing are discussed at the end of this chapter.

2.1 Routing Issues

The features of MANETs, such as node mobility and resource constraints, led to defining a set of desirable characteristics that a routing protocol should have to optimize the limited resources, and cope with dynamic topologies. Based on the qualitative and quantitative independent metrics given by Macker and Corson in [MC98], Stojmenovic *et al.* in [GSB03] define a set of metrics that a routing protocol for MANET should meet. Some of the those desirable properties are given below.

Loop-freedom: To avoid timeout or memorizing past traffic, the routing protocols for MANETs should be inherently loop-free. Also, node mobility might cause unexpected loops during packet forwarding. A routing protocol must consider the above mentioned situations to successfully deliver packets.

Distributed operation: The routing algorithms for MANETs can be classified into three different groups based on the criteria that a node follows to forward the packets. In a global routing algorithm, each node has knowledge of the entire network, which in turn makes

the routing task equivalent to the shortest path problem, if hop count is used as the main performance metric. On the other hand, in a localized routing algorithm, each node makes the decision of which neighbor to forward the message to based solely on the location of itself, its neighboring nodes, and destination. Between the two extremes is the zonal approach, where the network is divided into zones, with a localized algorithm applied within each zone, and a shortest path or other scheme applied for routing between zones [JNL99]. It is preferred to use localized algorithms if their performance is almost same as the performance of non-localized ones.

Path strategy: The number of paths followed by a routing protocol can be single or multiple. For instance, the shortest path route is an example of a single-path strategy, whereas flooding follows multiple paths to deliver the packets to the destination. The latter approach consumes a huge amount of extra bandwidth. Arguably, the ideal localized algorithm should follow a single path.

Metrics: Most routing schemes in MANETs use hop count as the metric, where hop count is the number of transmissions on a route from a source to the destination.

Memorization : Mobile devices don't have plenty of resources to memorize past traffic even though this may give the best result in routing. In addition, solutions that require nodes to memorize a route or past traffic are sensitive to changes in node activity and node mobility while routing is ongoing. It is better to design an algorithm that does not strictly depend on the past traffic at any node to forward the packet.

Guaranteed message delivery: Delivery rate is the ratio of the number of messages received by the destination to the number sent by the sender [BMJ+98]. The primary goal of every routing scheme is to deliver messages. However, the best routing scheme should be able to assure guaranteed message delivery. The guaranteed delivery property assumes the application of an ideal and collision-free medium access scheme.

Scalability: In MANETs, we call a routing algorithm scalable if it performs well with an arbitrary number of nodes. In [GSB03] the authors give a simple definition of scalability; a routing scheme is scalable if it is loop free, localized, and single-path.

Robustness: Although it is advantageous to use nodes' position for routing in mobile ad hoc networks, it imposes problems in terms of reliability: in particular, the accuracy of destination's position is an important problem to consider. A robust and scalable routing algorithm must, by design, be able to cope with the network dynamicity, which would help in dealing with a deviation on the node's position, thus avoiding the high mobility tracking

overhead incurred if it assumes its destination position. Such an algorithm should also be able to deliver messages when the communication model deviates from a unit disk graph, due to obstacles or noise. We explore one such model described in [BFNO01].

Fast adaptability to link changes: The ad hoc network is dynamic in nature, which demands the routing protocol to adapt to changes during routing to avoid packet dropping or an unexpected loop.

Unidirectional Link support: The probability of the presence of a unidirectional link in a mobile ad hoc network is higher than that in a wired network. The protocol design should consider the presence of a unidirectional link between nodes, otherwise it might cause partitioning in the network. For example, in some cases where two unidirectional links in opposite directions form the only means of communication between two ad hoc zones, the unidirectional link support would prevent unnecessary network partitioning.

Security: The protocol should be robust enough to survive under security attacks like spoofing, masquerading, replay attack and denial-of-service attack.

2.2 Classification of Routing Algorithms

In the last few years, a huge class of routing protocols for MANETs has been proposed in the literature either by modifying Internet routing protocols, or proposing new routing approaches. MANET routing protocols are typically subdivided into two main categories: *proactive* routing protocols and *reactive* routing protocols. Proactive routing protocols are derived from legacy Internet distance-vector and link-state protocols. These protocols try to maintain consistent and updated routing information for every pair of nodes in the network by propagating routing information periodically. They are sometimes referred to as *table-driven* protocols since the routing information is usually maintained in tables. Reactive or on-demand routing protocols, on the other hand, establish the route to a destination only when it is needed. The source node usually initiates the route request though the route discovery process if it is not available in the routing table. Then it maintains the route until either the destination becomes inaccessible, or the route is no longer used or expired [CCL03]. We describe some of the previously proposed protocols below.

2.2.1 Proactive Routing Algorithms

The main characteristic of these protocols is that each node maintains the route information for all the other nodes in the network. Route creation and maintenance are performed periodically or when a link is broken. Some of the widely used protocols in this group are: Destination-Sequenced Distance-Vector (DSDV) [PB94], Clusterhead Gateway Switched Network (CGSR) [MGLA96], Optimized Link State Routing (OLSR) [JMQ+03], and Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) [BOT01]. We will briefly describe the first two algorithms.

The Destination-Sequenced Distance-Vector (DSDV) protocol is a table-driven algorithm based on the classical Bellman-Ford routing mechanism with the properties needed in MANETs. Every node maintains a routing table with one route entry for each destination in which the minimum number of hops to reach that destination is recorded. Also, each record has a sequence number assigned by the destination node. The higher the sequence number the more recent the route is, that is, a node increases the sequence number if there is a change in its neighborhood. In this way a node can avoid loops by avoiding stale routes. Routing information is updated periodically, which consumes a huge amount of network bandwidth even though it provides the best route. To avoid unnecessary bandwidth wastage because of route updates, DSDV uses two types of packets, *full dump* with all the routing information and *incremental* with only the recent updates after the last full dump [RT99, CCL03].

CGSR extends DSDV by clustering the mobile nodes into small groups called cluster, and achieves a framework for routing, bandwidth allocation, and channel access. The whole network is divided into small clusters of nodes, and each cluster is controlled by a node called clusterhead. There also exist some nodes which are within the communication range of two or more clusterheads: they are called gateways. The sender first sends the packet to its clusterhead, the packet is then passed from clusterhead to clusterhead through the gateways until it arrives at the clusterhead of the destination node. However, frequent node movements might invoke the clusterhead selection algorithm frequently, which will keep the nodes busy selecting the clusterhead rather than routing. To avoid this overhead, a Least Cluster Change (LCC) clustering algorithm is introduced, so that the cluster heads only change when two of them come into contact, or when a node moves out of contact of all other cluster heads. Each node needs to maintain two tables, 'cluster member table' and 'routing table'. The first one records the clusterheads of all nodes, and the second one

keeps route information to reach the destination [RT99].

Most proactive routing protocols, such as DSDV and CGSR, are based on distance-vector-based shortest-path algorithm, and have the same degree of complexity during link failures and additions. However, they have different numbers of routing tables since they maintain routing information using different strategies.

2.2.2 Reactive Routing Algorithms

These protocols differ from proactive protocols in the way nodes maintain routes. Instead of maintaining routes for all the nodes, this type of routing algorithm creates a route only when it is needed by the source node. A node initiates a *route discovery* process within the network when it requires a route to a destination. The process terminates when it discovers a route or when all possible route permutations have examined. Once a route has been established, it is maintained by a *route maintenance* procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired [RT99]. Representative reactive routing protocols include: Dynamic Source Routing (DSR) [JMHJ02], Ad hoc On-Demand Distance Vector (AODV) [PR99], Temporally Ordered Routing Algorithm (TORA) [PC01], Associativity Based Routing (ABR) [Toh97], and Signal Stability Routing (SSR) [DRWT97]. We will briefly describe the two most widely used protocols DSR and AODV.

AODV is an enhancement of the DSDV protocol, which minimizes the required number of broadcasts by creating on-demand routes instead of maintaining a complete list of routes as in DSDV. The protocol has two major phases: *route discovery* and *route maintenance*. A source node initiates the route discovery process if it has no route information about the destination in its cache. The process continues until a route to the destination is located, or fresh route information is received from an intermediate node. AODV needs symmetric links since the destination or an intermediate node send back the route information through the neighboring node from which it received the first copy of the route request message. Route entries are dynamically maintained by considering the movement of both source and intermediate nodes.

DSR is a loop-free, source-based, on-demand routing protocol, which also has two phases: *route discovery* and *route maintenance*. Each node maintains a route cache that contains the source routes learned by the node. The route discovery process is initiated only when a source node does not already have a valid route to the destination in its route

cache; entries in the route cache are continually updated as new routes are learned. Route maintenance is accomplished through the use of *route error* packets, which are generated at a node when the data link layer encounters a transmission problem, and through acknowledgments. Source routing is used for packet forwarding.

DSR has a potentially larger control overhead and memory requirement than AODV since each DSR packet must carry full routing information, whereas in AODV, packets only contain the destination address. On the other hand, DSR can utilize both asymmetric and symmetric links during routing, while AODV only works with symmetric links. Also, nodes in DSR maintain multiple routes to a destination in their cache, which is helpful for link failure. In general, both AODV and DSR work well in small to medium size networks with moderate mobility [CCL03].

Comparison Between Proactive and Reactive Protocols

In general, on-demand reactive protocols are more efficient than proactive ones. On-demand protocols minimize control overhead and power consumption since routes are only established when required. In contrast, proactive protocols require periodic route updates to keep information current and consistent. In addition, they maintain multiple routes that might never be needed, and hence introduce unnecessary routing overheads.

Proactive routing protocols provide better quality of service than on-demand protocols. As routing information is constantly updated in the proactive protocols, routes to every destination are always available and up-to-date, and hence end-to-end delay can be minimized. However, in on-demand protocols, the source node has to wait for the route to be discovered before the actual routing. This latency in route discovery might be intolerable for real-time communications. The above considerations imply that proactive protocols are suitable for small scale static networks, while reactive protocols can normally work well in medium size networks with moderate mobility [CCL03, Roy03].

2.2.3 Hybrid Protocols

In addition to proactive and reactive protocols, another class of unicast routing protocols that can be identified is hybrid protocols. The Zone-Based Hierarchical Link State Routing Protocol (ZRP) [HP97] is an example of a hybrid protocol that combines both proactive and reactive approaches to bring together the advantages of the two approaches. ZRP defines

around each node a zone that contains the neighbors within a given number of hops from the node. Proactive and reactive algorithms are used by the node to route packets within and outside the zone, respectively.

2.2.4 Position-Based Routing

Position-based routing algorithms are localized, that is, distributed in nature, where simple local behavior achieves a desired global objective. In such algorithms, each node makes a decision about which neighbor to forward the message based solely on the location of itself, its neighboring nodes, and the destination.

Although routing-table-based solutions merely keep the best neighbor information on a route toward the destination, the communication overhead for maintenance of routing tables due to node mobility and topology changes is quadratic in network size. On the other hand, position-based algorithms do not require route establishment and maintenance, hence these algorithms may efficiently utilize the scarce storage resources and the relatively low computational power available to the wireless nodes. More importantly, given the numerous changes in topology expected in ad-hoc networks, no recomputation of the routing tables is needed and therefore we expect a significant reduction in the overhead. These features allow position-based routing protocols to quickly adapt to route changes, and they are more scalable than unicast protocols such as AODV, DSDV, and DSR [GSB03]. Localized routing is also uniform, in the sense that all the nodes execute the same protocol when deciding to which other node to forward a packet [CCL03]. These routing algorithms are classified in different ways in the literature. We will describe some of those related to our work.

Progress-Based Routing

In this type of strategy a node tries to forward the packet to one of its neighbors that is closest to the destination. Stojmenovic *et al.* [GSB03] distinguish these algorithms based on distance, progress, and direction. Since the main goal of these algorithms is to move towards the destination, we call all these algorithms together as progress-based algorithms. We will review some of these algorithms in this section.

The *Most Forward within Radius* (MFR) [TK84] routing algorithm maximizes the progress towards the destination by forwarding the packets to one of the neighboring nodes, whose projection onto the line between the current node and the destination is closest to

the destination. However, it also consumes maximum power to cover maximum distance, which in turn increases collisions with other nodes. Hence, instead of forwarding packets to the farthest neighboring node, the Nearest with Forward Progress (NFP) [HL86] scheme sends the packet to the node closest to the sender. The transmission can thus be accomplished with minimum power; hence the interference with the other nodes is minimized, while the probability of a successful transmission is maximized [GSB03].

In *Greedy* routing [Fin87, SL01], a node forwards a packet to its neighbor which is closest to the destination. The difference between *Greedy* and *MFR* routing is illustrated in Figure 2.2 which is taken from [SL01]. Finally, *Compass* or directional routing [KSU99] moves the packet to a neighboring node such that the angle formed between the current node, the next node, and the destination is minimized.

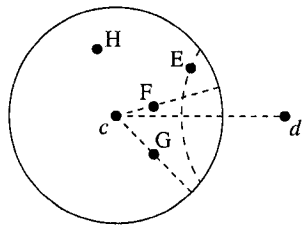

Figure 2.1: GREEDY chooses E, and COMPASS chooses F as next node.

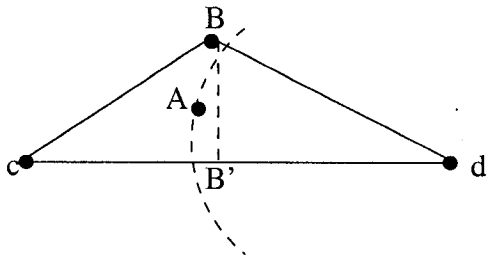

Figure 2.2: GREEDY chooses A, which is closer to the destination d, but MFR chooses B, whose projection on the $\overline{cd}$ line covers more distance to the destination.

In most cases, *MFR* and *Greedy* require the same number of hops to reach the destination. However, *Compass* routing needs more hops than *Greedy* routing, while the delivery rate is similar. All these methods have high delivery rates for dense graphs, but low delivery rate for sparse graphs. However, the performance of *MFR* and *Greedy* routing come

closing to matching the path length given by the shortest path algorithm in case of successful delivery [GSB03]. Stojmenovic *et al.* propose variants of *MFR*, *Greedy*, and *Compass* (called *MFR*, *GEDIR*, and *DIR*, respectively) by using 2-hop neighbor information to increase the delivery rate. Also, they use partial flooding with *MFR* and *Greedy* algorithms to ensure guaranteed packet delivery [SL01, LLS01, LS03].

Drawbacks of Greedy and Compass Routing

The *Greedy* and *Compass* algorithms are known to fail to deliver the packet in certain situations, and their enhancements introduce complexity in routing. We will now explain two situations where these algorithms fail to deliver the packet. After that we will describe another class of position-based routing that deals with the shortcomings of above mentioned algorithms, and provides an alternative solution that guarantees packet delivery.

In greedy routing, a packet may get stuck at a node during routing because it encounters a local maximum, that is, a node that has no neighbor closer to the destination than itself, or two adjacent nodes that are equally close to the destination, where none of their other neighbors are closer. A simple example of such a scenario is given by Karp and Kung in [KK00], and reproduced in Figure 2.3. Here, s is closer to the destination d than its two

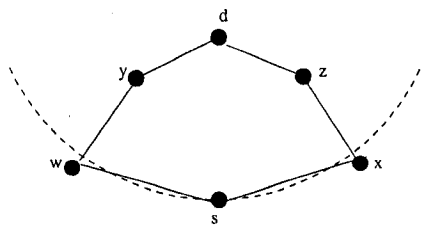

Figure 2.3: An example where *Greedy* routing fails because s is a local maximum in its geographic proximity to d.

neighbors w and x, where the dashed arc about d has a radius equal to the distance between s and d. Although the paths, $(s \rightarrow x \rightarrow z \rightarrow d)$ and $(s \rightarrow w \rightarrow y \rightarrow d)$ exist to d, s will not be able to forward the packet to w or x using greedy routing. Here s is a local maximum in its proximity to d [KK00].

Also, *Compass* routing can get into an unavoidable loop, which can be illustrated by the example in Figure 2.4 given by the authors in [GSB03]. Let the loop consist of four nodes E, F, G, and H in an unit disk graph. Let the source be any node in the loop, say E. E

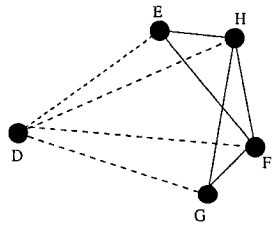

Figure 2.4: An example where COMPASS routing face an unavoidable loop.

forwards the packet to F since it forms the smallest angle $\angle DEF$ among all the neighbors of E. Similarly, F selects G, G selects H, and H selects E [GSB03].

Algorithms with Guaranteed Delivery

By combining the above progress-based rules for the choice of the next neighbor and to exit from a local maximum, several routing algorithms have been defined. Stojmenovic *et al.* call this class of routing *nearly stateless routing with guaranteed delivery* in [GSB03]. In [BMSJ99], the authors propose *face routing*, which is based on the so called *right hand rule* for exploring a maze. The rule states that if a player in a maze walks around never lifting her right-hand from the wall, then she will eventually visit every wall in the maze. More specifically, if the constructed maze define the faces of a connected planar straight line graph, the player will visit every edge and vertex of the face. Bose *et al.* use a planar and connected subgraph of a unit disk graph called the *Gabriel Graph (GG)* (we will define the *GG* in the next section). Once the Gabriel graph is extracted from the network, routing is performed along its edges. Its planarity and connectivity ensure message delivery by routing along the faces of the graph that intersect the line between the source and the destination. A rough idea of face routing is shown in Figure 2.5, which is taken from [BFNO01].

Bose *et al.* in [BMSJ99] also found that there is a trade-off between the delivery rate and the stretch factor if we only use face routing. Then they conclude that face routing alone is not very efficient, and suggest using it in conjunction with progress-based routing heuristics that do not guarantee delivery. In [BMSJ99], they combine greedy routing with face routing and called the resulting algorithm *Greedy-Face-Greedy (GFG)*. The algorithm first forwards the packet greedily until it reaches the destination or a local maximum. In the latter case the algorithm switches to face routing, and traverses the face until it reaches

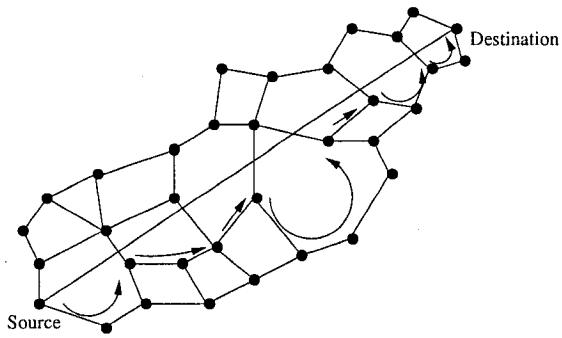

Figure 2.5: An example of face routing.

another node which is strictly closer to the destination than the node which was the local maximum. The algorithm then again switches back to the greedy mode at this point. The authors also show that the greedy algorithm alone has a low delivery rate, but *GFG* achieves 100% delivery rate at the price of losing the impressive stretch factor of the greedy algorithm [BMSJ99].

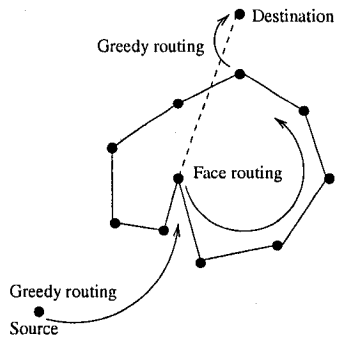

Figure 2.6: Alternation of *Greedy* and *Face* routing.

Karp and Kung in [KK00], transformed the *GFG* algorithm into the *Greedy Perimeter Stateless Routing (GPSR)* protocol by including discussion of the IEEE 802.11 medium access control scheme. They evaluate the performance of their algorithm against *DSR*, and show that their algorithm is more scalable than source based protocols under conditions of mobility.

Although the algorithm in [BMSJ99] ensures message delivery, it is likely to fail if there is any instability in the transmission ranges of the mobile nodes. Instability in the

transmission ranges means that the area a mobile node can reach is not necessarily a disk and the range may vary between $r = (1 - \varepsilon)R$ and R, for some $\varepsilon > 0$. In [BFNO01] the authors consider such instability, and propose this model as a generalization of the *UDG*. They are able to handle unstable situations, where nodes may or may not communicate directly because of obstacles (buildings and bad weather), with their proposed model.

Randomized Position-Based Routing

Another simple and efficient way to avoid local maxima and loops is to use randomization. A position-based routing algorithm is *randomized* if the next neighboring node is chosen randomly out of the neighbors of the current node. In [BM99], Bose and Morin proposed a randomized algorithm called *Random Compass* in the context of triangulations (they refer to position-based routing as *memoryless* and *online* routing). In *Random Compass* the next node is chosen uniformly at random from the two nodes that satisfy the directional heuristic going in the clockwise and the anticlockwise directions. The algorithm has a higher delivery rate than the deterministic algorithm at the price of a higher stretch factor. In [BM99] and [BBC+02a] Bose *et al.* proved that *Random Compass* guarantees packet delivery for all triangulations, and all convex subdivisions, respectively.

2.2.5 Directed Flooding-Based Routing

Directed flooding is another class of position-based routing where nodes forward the packets to all neighbors that are located in the direction of the destination. DREAM [BCSW98] and *Location-Aided Routing (LAR)* [KV98] are two routing algorithms that apply this principle. LAR uses directed flooding only for route discovery, while DREAM applies a restricted flooding for packet delivery.

Location-Aided Routing (LAR) decreases overhead of route discovery by utilizing information about the location of the mobile hosts. The authors proposed two protocols which use location information to reduce the search space for a desired route, and hence reduce the number of route discovery messages. The search for a route is limited to the so-called request zone, determined based on the expected location of the destination node at the time of route discovery. LAR floods the network, sending packets only inside the request zone [KV98, CCL03].

In DREAM, mobile nodes broadcast their location information throughout the network periodically. As a result, the source knows the up-to-date location of the destination before

forwarding packets. Similar to the LAR approach, the source node defines an expected region for the destination based on the location of destination. The expected region is a circle centered on the last known receiver location, which represents the area where the receiver should be, taking into account the node mobility from its last known position. A packet is then forwarded towards the expected region [CCL03].

2.2.6 Hierarchical Routing

Hierarchical routing algorithms are a combination of position and non-position-based routing algorithms. Position-based routing is typically used for long distances, while a non-position-based algorithm is used at the local level. *Grid* routing [LTS01] and the *Terminode* routing [BBC+01] are hierarchical routing protocols in which routing is structured in two layers. Location-aware routing is used for routing on long distances, and when a packet arrives close to the destination a proactive distance vector scheme is used [CCL03].

In GRID routing the authors partition the geographic area into a number of squares called grids. In each grid, if not empty, one mobile node is elected as the leader of the grid. Routing is then performed in a grid-by-grid manner through those leaders. The size d of each grid depends on transmission radius of the node. One leader is able to communicate directly with leaders in neighboring grids, and all nodes within each grid are connected to their leaders [GSB03].

The Terminode routing scheme is a combination of two protocols called *Terminode Local Routing (TLR)* and *Terminode Remote Routing (TRR)*. The TLR protocol is used when destinations are in the vicinity of a terminode without considering location information. However, TRR is used to send data to remote destinations and uses geographic information. The authors claim that the combination of these two approaches help Terminode to achieve scalability.

2.3 Network Topology for MANET

In this section we define some of the properties that the network topology should have, especially for routing in MANETs. Also, we present some of the widely used network topologies for routing.

It is common to separate the network design problem from the management and control of the network to simplify these two tasks. However, the decisions made at the network

design phase may strongly affect the network management and control phase. In particular, if the issue of designing efficient routing schemes is not taken into account by the network designers, then the constructed network might not be suited for supporting a good routing scheme. For example, a backbone-like network topology is more suitable for a hierarchical routing method than a flat network topology [Li03b, Li03a]. In this section, we focus on the network topologies of ad hoc networks in the context of position-based routing.

Network connectivity, bounded node degree, low-stretch factor, robustness to mobility, and linear number of links are the desirable properties of the network topology of mobile ad hoc networks to efficiently support routing. Due to node mobility and resource constraints, it is challenging to design a network topology for wireless ad hoc networks, that is suitable for designing an efficient routing scheme. It is also preferred that the underlying network topology be constructed in a localized manner because of node mobility and resource constraints.

2.3.1 Preliminaries

This section presents some of the graph theoretic and geometric terminology used in this thesis.

A *proximity graph* on a finite set $P \subset \Re^2$ is obtained by connecting pairs of points of P with line segments if the points are considered to be *close* in some sense. Different definitions of closeness give rise to different proximity graphs. One technique for defining a proximity graph on a set of points is to select a geometric region defined by two points of P – for example the smallest disk containing the two points – and then specifying that a segment is drawn between the two points if and only if this region contains no other points from P. Such a region will be referred to as a *region of influence* of the two points [BDEK02].

2.3.2 Properties of the Topology

For efficient routing we need a topology which is a subgraph of a UDG, with the following desirable properties. The detailed description of these properties can be found in [ALW+03, Raj02].

Efficient Localized Construction: A distributed algorithm constructing a graph G is a localized algorithm if every node u can exactly decide all edges incident on u based only on the information of all nodes within a constant number of hops of u. More precisely, the number of messages sent by each node should be bounded. The message complexity at each node can be $O(d\log d)$ to construct the underlying topology efficiently, where d is the number of one-hop neighbors. It is obvious that the underlying network topology for MANETs should be constructed and maintained in a localized manner due to the limited resources of the wireless nodes.

Connectivity: Another crucial, and perhaps, the most basic requirement of a topology is that it should be connected. More precisely, we require that any two nodes that are connected in the underlying UDG are also connected in the extracted topology.

Spanner: Let $G = (S, E)$ be an n-vertex connected weighted graph over S. The distance in G between two vertices $(u, v) \in S$ is the total weight of the shortest path between u and v and is denoted by $d_G(u, v)$. A subgraph $H = (S, E')$, where $E' \subseteq E$, is a *t-spanner* of G if for every $u, v \in S$, $d_H(u, v) \leq t \cdot d_G(u, v)$. The value of t is called the *spanning ratio*, and referred to as *length spanning ratio*, *hop spanning ratio*, or *power spanning ratio* if the weight is the Euclidean length of the link, hop count, or power for communication, respectively.

Sparseness: The topology should be sparse in order to ensure efficient routing. A network topology is considered as sparse if it has only a linear number of links.

Bounded Node Degree: A node with bounded degree means it needs to maintain and process a constant number of neighbors. This is important since every wireless node has limited computational resources, storage, and limited power.

Planar: The network topology is called planar if there are no crossing edges. Some routing algorithms, such as *Face* routing, *Greedy-Face-Greedy (GFG)*, and *Greedy Perimeter Stateless Routing (GPSR)* need the network topology to be planar.

Robustness: An additional challenge in the design of the network topology for MANETs is to ensure some degree of robustness to the mobility of nodes. One measure of robustness of the topology is given by the maximum number of nodes that need to update their topology information as a result of the movement of a node. This number depends on the size of the neighborhood of a mobile node, and the relative location of the nodes.

2.3.3 Some Existing Network Topologies for MANET

This section presents a set of network topologies for ad hoc networks.

We have seen in Chapter 1 that an ad hoc network can be modeled as a UDG which is used as the network topology by many position-based routing algorithms, especially progress-based algorithms. The main advantage of this basic topology is that it can be constructed in a simple distributed manner. However, it cannot ensure bounded node degree and sparseness (might have $O(n^2)$ edges in the worst case), which could help less maintenance at each node. Moreover, the performance (in terms of packet delivery rate and stretch factor) of those algorithms is not satisfactory on UDGs. The algorithms with guaranteed delivery, on the other hand, need to use a planar subgraph of the UDG. Based on this motivation a number of distributed algorithms have been proposed to construct subgraphs of UDGs, which are proximity graphs. We shall review some of these graphs related to this thesis.

Yao Graph: The *Yao graph* (also called a *Theta graph* [BGM04]) $\vec{YG}_k(G)$ with an integer parameter $k \geq 6$ is defined as follows [Yao82]. At each node u, k equally-separated rays originating at u define k cones. Given an angle θ such that $\theta = \pi/k$, we define $Cone(u,\theta)$ to be the cone of angle θ in $disk(u,r)$. In each cone, only the directed edge (u,v) to the nearest neighbor v, if any, is part of $\vec{YG}_k(G)$. Ties are broken arbitrarily. The definition can also be written as follows.

$$\forall (w \neq u, v) \wedge (w, v \in Cone(u,\theta)) : d(u,v) < d(u,w)$$

The Yao graph has been re-discovered by several researchers for topology control in wireless ad hoc networks with directional antenna [Li03b]. For $G = UDG(S)$, let $YG_k(G)$ be the undirected graph obtained if the direction of each edge in $\vec{YG}_k(G)$ is ignored. Then, the spanning ratio of the $YG_k(G)$ is $1/(1-2\sin\pi/k)$ in terms of Euclidean distance [LWW01, Li03b, Li03a]. It is known that the node out-degree of the $\vec{YG}_k(G)$ with an integer parameter $k \geq 6$ is bounded by some positive constants, and it is connected if the underlying UDG is connected. However, this graph cannot guarantee a bounded node degree, that is, the node in-degree of the graph could be as large as $O(n)$ [LWW01]. They propose another sparse topology, *Yao and Sink*, that has both a constant bounded node degree and length spanning ratio. However, all these graphs related to the *Yao* graph are not planar graphs.

Gabriel Graph: Denote the closed disk containing the points u and v and with diameter

$dist(u,v)$ by $disk(u,v)$. Then the *Gabriel Graph* of G, denoted by $GG(G)$, is defined as follows. Given any two adjacent nodes u and v in G, the edge (u,v) belongs to $GG(G)$ if and only if no other node $w \in G$ is located in $disk(u,v)$. This can be also represented by the following equation.

$$\forall w \neq u,v : d^2(u,v) < [d^2(u,w) + d^2(v,w)]$$

The Gabriel Graph is used as a planar subgraph for *Face* routing, *GFG*, and the *GPSR* routing algorithms. The Euclidean length spanning ratio of $GG(G)$ is $(4\pi\sqrt{2n-4})/3$ in the worst case [BDEK02], and it is known that the $GG(G)$ is connected if the underlying *UDG* is connected [BMSJ99]. Since *GG* has large length stretch factor in the worst case, some other structure is needed if we want to bound the distance traveled by a routing algorithm from the source to the destination. One good candidate in this context is the *Delaunay triangulation* given by the set of nodes.

Delaunay Triangulation: Assuming no four vertices in S are co-circular, a triangulation[1] of a set of points S is a Delaunay triangulation (DT) if the circumcircle of each of its triangles does not contain any other nodes of S in its interior. More precisely, if we define a closed $disk(u,v,w)$ as the circumcircle of the $\triangle uvw$, then for all $x \neq u,v,w \wedge x \in S$ the $disk(u,v,w)$ should be empty of x.

Bose *et al.* present an interesting history of the study of DT in [BDEK02], which is given below. First, Chew [Che86, Che89] showed that *DT* is a $\pi/2$-spanner. Then, Dobkin *et al.* [DFS90] showed that it is a $((1+\sqrt{5})/2)\pi \approx 5.08$-spanner. Finally, Keil and Gutwin [KG89, KG92] improve this value to $2\pi/(3\cos\pi/6) \approx 2.42$. In addition, it is connected and planar. However, it is not appropriate for MANET environment because it cannot be constructed locally. Also, it might have edges longer than the transmission radius of the nodes. In [LCW02] the authors propose a localized algorithm to construct a graph, called Localized Delaunay graph $LDel^{(k)}(S)$, which is supergraph of $UDel(S)$. $LDel^{(k)}(S)$ is planar for any $k \geq 2$, but for $k = 1$ it might have crossing edges. Here, $UDel(S) = UDG(S) \bigcap Del(S)$, that is, $UDel(S)$ has no edges longer than the transmission radius.

Planarized Localized Delaunay triangulation : The triangle $\triangle uvw$ is called a *k-localized Delaunay triangle* if the interior of the circumcircle of $\triangle uvw$, denoted by $disk(u,v,w)$, does not contain any vertex of S (that is, a k-neighbor of u, v, or w), and all edges of the triangle

[1] A triangulation is a convex subdivision where each interior face is a triangle.

$\triangle uvw$ have length no longer than the transmission radius. The *k-localized Delaunay* graph over a vertex set S , denoted by $LDel^{(k)}(S)$, has exactly all unit (not longer than transmission radii) Gabriel edges and edges of all k-localized Delaunay triangles. If we generalize the definition of $LDel^{(k)}(S)$ only for one-hop neighbors, then the resulting graph is denoted by $LDel(S)$ (ignoring the superscript 1).

The basic approach of constructing $LDel(S)$ is as follows. Each node u computes the Delaunay triangulation $Del(N(u))$ of its 1-neighbors $N(u)$, including u itself. Node u then sends messages to its neighbors asking if the triangles in $Del(N(u))$ can be accepted to $LDel(S)$. Its neighbor v accepts the triangle if it is in $Del(N(v))$.

$LDel(S)$ may contain intersecting edges, that is, it does not ensure planar properties even though it can be decomposed into two planar graphs. Instead of decomposing $LDel(S)$ into two planar graphs, Li *et al.* present an efficient algorithm to extract a planar graph $PLDel(S)$ out of $LDel(S)$. The basic approach according to [LCW02] is as follows. Each node u gathers all the Gabriel edges and 1-localized Delaunay triangles incident on it. The node then checks if there is any intersection between two triangles, lets say $\triangle uvw$ and $\triangle xyz$, known by u. If they intersect such that the circumcircle of $\triangle uvw$ contains a vertex from $\triangle xyz$, u removes $\triangle uvw$ from its list. All the remaining triangles are kept in the list if corresponding neighbors also agree to keep them.

The resulting planar graph is called $PLDel(S)$, and is a t-spanner of a unit-disk graph in terms of Euclidean length, where $t = (4\sqrt{3}/9)\pi \approx 2.42$. Also, it ensures connectivity, if the underlying UDG is connected, with a bounded node degree and linear number of edges. The total communication cost to construct the $PLDel(S)$ in a distributed fashion is $O(n)$, and the computation cost at each node is $O(d\log d)$, where d is number of 1-hop neighbors of each node.

Restricted Delaunay Graph: Gao *et al.* in [GGH+01] proposed another structure, called the Restricted Delaunay graph (RDG), which can be defined as follows. A Restricted Delaunay Graph of a set of points in the plane contains all the Delaunay edges with length at most transmission radii of the nodes. In other words, they call any planar graph containing $UDel(S)$ as a restricted Delaunay graph. It is known that RDG is a planar t-spanner in terms of Euclidean length with $t = (1 + \sqrt{5}/2)\pi \approx 5.08$. It is also connected if the underlying UDG is connected, and can be constructed and maintained in a distributed manner. The communication cost to construct the RDG is $O(n^2)$, and the computation cost at each node is $O(d^3)$, where d is the number of its 1-hop neighbors [LCW02].

All of the above mentioned subgraphs of UDG have high adaptability, since a change in a node location will only require the nodes in its neighborhood to recompute their edges in the topology. The definitions of the above mentioned graphs are diagrammatically given below. Also, different network topologies for the same set of nodes as presented by Li *et al.* in [Li03b] are given next. Table 2.1 presents the salient features of all the topologies discussed above.

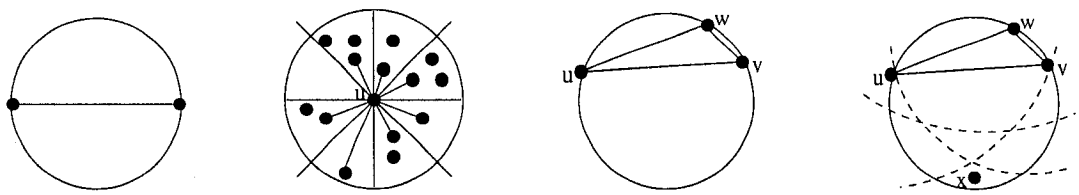

Figure 2.7: Definition of different graphs.

Topological Properties	YAO	GABRIEL	PLDEL	RDG	DT
Distributed Construction	Yes	Yes	Yes	Yes	No
Connectivity	Yes	Yes	Yes	Yes	Yes
Planarity	No	Yes	Yes	Yes	Yes
Spanning Ratio (Euclidean length)	$\frac{1}{1-2\sin\pi/k}$	$\frac{4\pi\sqrt{2n-4}}{3}$	$\frac{4\sqrt{3}}{9}\pi$	$\frac{1+\sqrt{5}}{2}\pi$	$\frac{4\sqrt{3}}{9}\pi$

Table 2.1: The properties of existing network topologies.

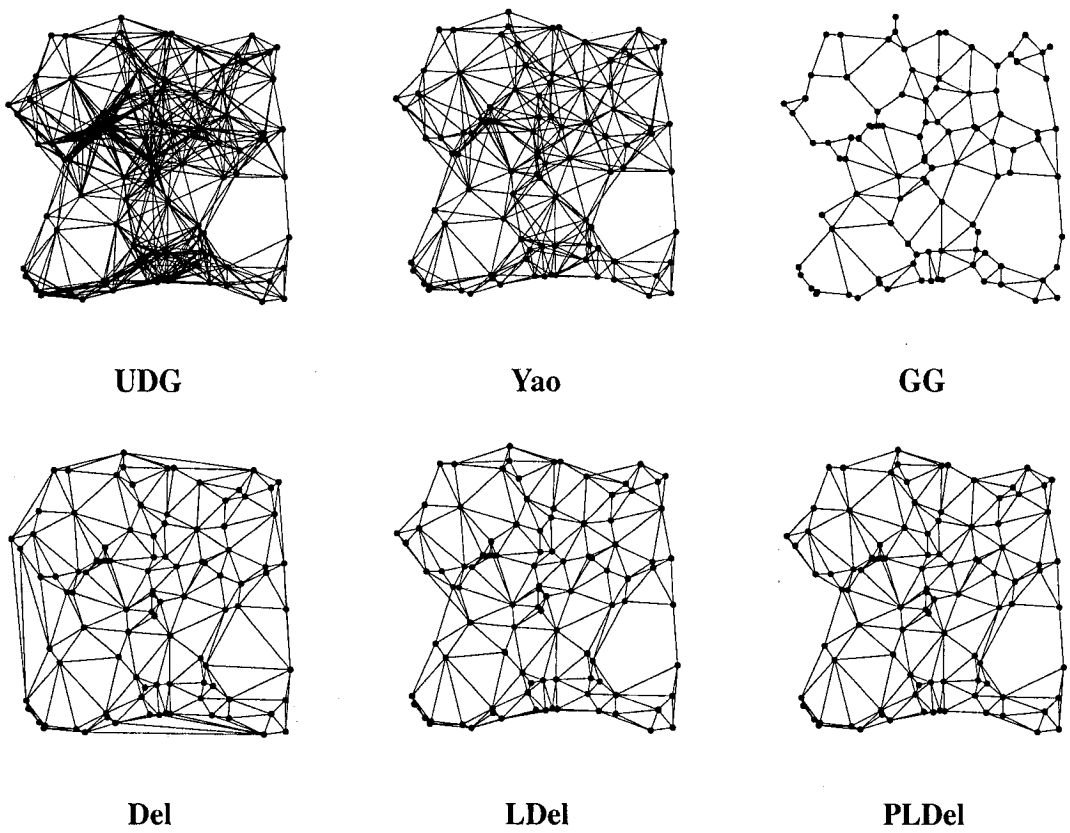

Figure 2.8: Different network topologies taken from [Li03b].

Chapter 3

Randomized Routing Algorithms

In this chapter we first define the network model for our randomized algorithms, and then classify our randomized algorithms based on the choice of the next candidate nodes. Finally, we describe the properties of our algorithms.

3.1 The Network Model and Other Definitions

We model a MANET according to the model given in [BFNO01]. A set of mobile hosts are spread out in an environment that is modeled by the Euclidean plane: each mobile host with $x-$ and $y-$coordinates is represented by the point (x,y). All distances are Euclidean distances in the plane. We will use the following hypothesis and notation:

1. Any mobile host knows the coordinates (x,y) of its position.
2. The transmission range of each mobile host is r, that is, two hosts can directly communicate with each other if their distance is at most r.
3. Each mobile host has an omni-directional antenna which covers a circular area of radius r.
4. Communication links are *bidirectional*, that is, if a mobile host u is able to receive signals from a mobile host v, then v is also able to receive signals from u.

Based on the above hypothesis, we can represent a MANET as a geometric undirected graph $G = (S,E)$, where vertices represent mobile hosts and edges represent a link through which a pair of mobile hosts can communicate directly. The set of vertices S is thus a set

of points in the Euclidean plane. Let $d(u,v)$ be the distance between the points u and v in the plane. The set of edges E $\{\{u,v\} : u,v \in S, d(u,v) \leq r\}$, that is, E contains all the pairs of mobile hosts at a distance of at most r [BFNO01]. The resulting graph $UDG(S)$ is called a unit disk graph. For node u, we denote the set of its neighbors by $N(u)$. A *convex subdivision* is an embedded planar graph such that each face of the graph is a convex polygon, except the outer face which is the complement of a convex polygon.

Given a unit disk graph $UDG(S)$ corresponding to a set of points S, and a pair (s,d) where $s,d \in S$, the problem of online position-based routing is to construct a path in $UDG(S)$ from s to d, where in each step, the decision of which node to go to next is based only on the coordinates of the current node c, $N(c)$, and d. Here, s is termed the source and d the destination. Frequently, we will also refer to the line $\overline{cd}$ passing through c and d.

An algorithm is *deterministic* if, when at c, the next node is chosen deterministically from $N(c)$, and is *randomized* if the next step taken by a packet is chosen randomly from $N(c)$.

The routing algorithm may or may not succeed in finding a path from s to d. We use the following notion of a graph defeating an algorithm from [BM99]. A deterministic algorithm is *defeated* by a graph $G = (S,E)$ if there is a pair $(s,d) \in S$ such that a packet using the algorithm never reaches the destination d when beginning at the source s [BM99]. A randomized algorithm is defeated by a graph $G = (S,E)$ if there is a source/destination pair $(s,d) \in S$ such that a packet using the algorithm and originating at source s has probability 0 of reaching destination d in any finite number of steps.

We are interested in the following performance measures for routing algorithms: the *delivery rate*, that is, the percentage of times that the algorithm succeeds, and the *stretch factor*, the average ratio of the length of the path returned by the algorithm to the length of the shortest path in the graph. Here the length of the path is taken to mean the number of hops in the path.

Given a node u, we denote the closed disk centered at node u with radius ℓ by $disk(u,\ell)$. Given an angle θ such that $0 \leq \theta \leq 2\pi$, we define $Sector(c,d,\theta)$ to be the closed sector given by angle θ in $disk(c,r)$ that is bisected by the line segment $\overline{cd}$. Further, given an α such that $0 \leq \alpha \leq 1$, we define *Periphery*, *Core*, and *Wing* as follows. Also, Figure 3.1 illustrates the given definitions.

- $Periphery(c,d,\theta,\alpha) = Sector(c,d,\theta) \cap disk(d,R)$ where R is chosen such that α is

the ratio of the area of $Periphery(c,d,\theta,\alpha)$ to the area of $Sector(c,d,\theta)$.

- $Core(c,d,\theta,\alpha) = Sector(c,d,\theta) - Periphery(c,d,\theta,\alpha)$.
- $Wing(c,d,\theta,\alpha) = (disk(c,r) \cap disk(d,R)) - Periphery(c,d,\theta)$.

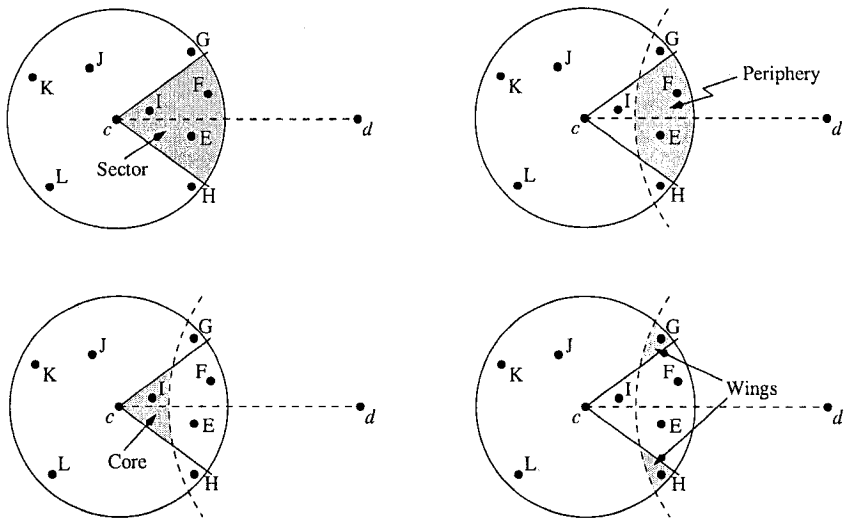

Figure 3.1: Illustrations of the definitions of *Sector*, *Periphery*, *Core*, and *Wing* (represented by the shaded regions). Note that $Sector(c,d,\theta) = Core(c,d,\theta,\alpha) \cup Periphery(c,d,\theta,\alpha)$. In this example, FARINSECTOR selects x uniformly at random out of nodes E and F, the only nodes in $Periphery(c,d,\theta,\alpha)$, while GREEDYINSECTOR picks $x = F$ since F is closest to d out of E, F, and I, the nodes in $Sector(c,d,\theta)$.

3.2 Classification of Randomized Routing Algorithms

We have classified our randomized algorithms into three different groups. The *Best-Two-Neighbors* routing algorithms are just the straightforward randomizations of the *Greedy* and *Compass* routing algorithms, where two best candidates are chosen out of the neighbors of the current node.The next node is then chosen uniformly at random out of those two neighbors. In the *Sector* algorithms a sector, centered at the line between the current node and the destination node, defines the set of candidate nodes. The next node is chosen randomly from the candidate nodes according to some probability distribution. Finally, the *AB* algorithms pick one neighbor of the current node *above* the line passing through the current node and the destination, and another neighbor *below* this line. Then the next node

is chosen randomly from these two neighbors according to some probability distribution. The exact choice of the candidate nodes and the probability distribution determine the algorithm.

3.3 Definition of the Algorithms

In this section we define our randomized algorithms. In what follows, we always assume that the current node is c, the next node is x, and the destination node is d. The algorithms below differ in how to choose x from among the set $N(c)$.

3.3.1 Best-Two-Neighbors Algorithms

The first group of algorithms we consider are straightforward randomizations of the greedy and directional heuristics.

BEST2GREEDY-U: Let n_1 and n_2 be the closest and second closest neighbors of c to the destination d. The next node is chosen uniformly at random out of these two nodes.

BEST2COMPASS-U: Let n_1 and n_2 be the neighbors of c such that $\angle dcn_1$ (or $\angle n_1cd$) and $\angle n_2cd$ (or $\angle dcn_2$) are the two smallest such angles among all neighbors of c. The next node is chosen uniformly at random out of these two nodes.

The behavior of the above mentioned algorithms is illustrated in Figure 3.2.

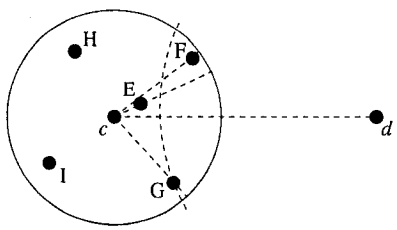

Figure 3.2: BEST2GREEDY-U sets $n_1 = F$ and $n_2 = G$ whereas BEST2COMPASS-U sets $n_1 = E$ and $n_2 = F$.

3.3.2 Sector Algorithms

In this section we describe several algorithms employing a sector-based approach, where the next node is chosen based on some probability distribution, from the candidate nodes

inside the defined sector.

FARINSECTOR: The next node x is chosen uniformly at random from the first non-empty set in the following sequence: *Periphery(c,d,θ,α), Wing(c, d, θ,α), Core(c,d, θ, α), Sector(c, d,θ)*, where $0 \leq \theta \leq 2\pi$. Appendix C presents the theoretical calculations necessary for the dynamic maintenance of a fixed size *Periphery*.

GREEDYINSECTOR: If $Sector(c,d,\theta)$ is not empty then x is chosen to be the neighbor of c with minimum distance from d, in $Sector(c,d,\theta)$. Otherwise, x is chosen uniformly at random from the set $N(c)$.

RANDOMINSECTOR: The next node x is chosen uniformly at random from the first non-empty set in the following sequence: $Sector(c,d,\theta_1)$, $Sector(c,d,\theta_2)$, $N(c)$, where $0 \leq \theta_1 \leq \theta_2 \leq 2\pi$.

The behavior of the above mentioned algorithms is illustrated in Figures 3.1 and 3.3.

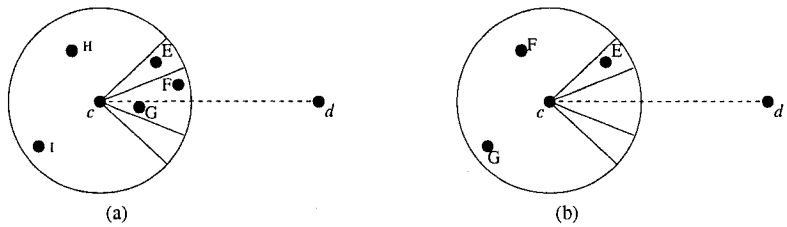

Figure 3.3: In (a) GREEDYINSECTOR selects F as next node out of E, F, and G, whereas RANDOMINSECTOR picks x uniformly at random out of F and G. In (b) RANDOMINSECTOR selects E as the next node since $Sector(c,d,\theta_1)$ is empty and E is the only node in $Sector(c,d,\theta_2)$.

WEIGHTEDINSECTOR: For each node n_i in $Sector(c,d,\theta)$, let $a_i = \angle n_i cd$, and let its weight be $w_i = 1/a_i$, and the total weight be $W = \sum w_i$. The next node x is chosen with probability w_i/W out of these candidate nodes if the sector is not empty. Otherwise, x is chosen uniformly at random from the set $N(c)$.

WEIGHTEDSECTOR: For each node n_i in $N(c)$, let $a_i = \angle n_i cd$ be the angle, and let its weight be $w_i = 1/a_i$, and the total weight be $W = \sum w_i$. The next node x is chosen with probability w_i/W from the set $N(c)$.

The behavior of the above mentioned algorithms is illustrated in Figures 3.4.

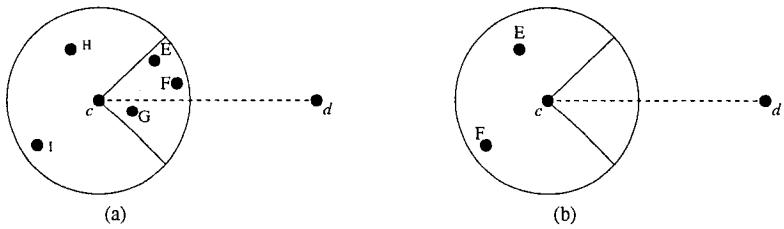

Figure 3.4: In (a) both WEIGHTEDINSECTOR and WEIGHTEDSECTOR select G as the next node with higher probability. In (b) WEIGHTEDINSECTOR selects x uniformly at random out of F and G, whereas WEIGHTEDSECTOR sets E as the next node with higher probability.

3.3.3 AB Algorithms

In this section we propose another class of randomized algorithms, called the *AB* algorithms. The idea behind these algorithms is to choose two candidate nodes: one from above the $\overline{cd}$ line and the other from below that line. The candidate neighbors are chosen according to the greedy and directional heuristics. The next node is then chosen either uniformly at random from the two neighbors or according to a biased coin where the bias is determined by distance from current node to destination, or angle formed by the neighbor node, the current node, and the destination.

AB:COMPASS-U: Let n_1 be the neighbor of c from above the $\overline{cd}$ line such that $\angle dcn_1$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below the $\overline{cd}$ line such that $\angle n_2cd$ is the smallest among all such neighbors. The next node x is chosen uniformly at random from n_1 and n_2.

AB:COMPASS-A: Let n_1 be the neighbor of c from above the $\overline{cd}$ line such that $\theta_1 = \angle dcn_1$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below the $\overline{cd}$ line such that $\theta_2 = \angle n_2cd$ is the smallest among all such neighbors. The next node x is chosen from n_1 and n_2 with probability $\theta_2/(\theta_1+\theta_2)$ and $\theta_1/(\theta_1+\theta_2)$, respectively.

AB:COMPASS-D: Let n_1 be the neighbor of c from above the $\overline{cd}$ line such that $\angle dcn_1$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below the $\overline{cd}$ line such that $\angle n_2cd$ is the smallest among all such neighbors. Let $dis_1 = dist(n_1,d)$ and $dis_2 = dist(n_2,d)$. The next node x is chosen from n_1 and n_2 with probability $dis_2/(dis_1+dis_2)$ and $dis_1/(dis_1+dis_2)$, respectively.

The behavior of the above mentioned algorithms is illustrated in Figure 3.5.

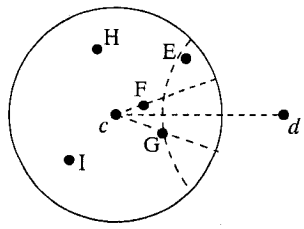

Figure 3.5: AB:COMPASS algorithms choose $n_1 = F$, $n_2 = G$ as the candidate nodes, and AB:COMPASS-U will choose either n_1 or n_2 with equal probability, whereas AB:COMPASS-D and AB:COMPASS-A will choose n_2 and n_1 with higher probability than the other node, respectively.

AB:GREEDY-U: Let n_1 be the neighbor of c above the $\overline{cd}$ line such that the distance $dist(n_1,d)$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below $\overline{cd}$ such that the distance $dist(n_2,d)$ is the smallest among all such neighbors. The next node x is chosen uniformly at random from n_1 and n_2.

AB:GREEDY-A: Let n_1 be the neighbor of c above the $\overline{cd}$ line such that $dist(n_1,d)$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below $\overline{cd}$ such that the distance $dist(n_2,d)$ is the smallest among all such neighbors. Let $\theta_1 = \angle n_1cd$ and $\theta_2 = \angle n_2cd$. The next node x is chosen from n_1 and n_2 with probability $\theta_2/(\theta_1+\theta_2)$ and $\theta_1/(\theta_1+\theta_2)$, respectively.

AB:GREEDY-D: Let n_1 be the neighbor of c above the $\overline{cd}$ line such that $dis_1 = dist(n_1,d)$ is the smallest among all such neighbors. Similarly, n_2 is the neighbor of c below $\overline{cd}$ such that $dis_2 = dist(n_2,d)$ is the smallest among all such neighbors. The next node x is chosen from n_1 and n_2 with probability $dis_2/(dis_1+dis_2)$ and $dis_1/(dis_1+dis_2)$, respectively.

The behavior of the above mentioned algorithms is illustrated in Figure 3.6.

Now we will define two deterministic AB algorithms, which are variations of the *Greedy* and *Compass* routing algorithms.

AB:GREEDY-A-DET: This deterministic algorithm is defined as is AB:GREEDY-A with the following difference. If $\theta_1 < \theta_2$, then the next node x is n_1, otherwise x is n_2.

AB:COMPASS-D-DET: This deterministic algorithm is defined as is AB:COMPASS-D with the following difference. If $dis_1 < dis_2$, then the next node x is n_1, otherwise x is n_2. This algorithm is similar to *Greedy-Compass* proposed by Bose *et al.* in [BBC+02b] except that we strictly choose at most one possible neighbor from each side of the $\overline{cd}$ line.

The behavior of the above mentioned algorithms is illustrated in Figure 3.7.

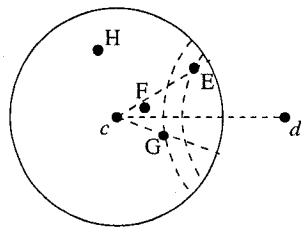

Figure 3.6: AB:GREEDY algorithms choose $n_1 = E$, $n_2 = G$ as the candidate nodes, and AB:GREEDY-U will choose either n_1 or n_2 with equal probability, whereas AB:GREEDY-D and AB:GREEDY-A will choose n_1 and n_2 with higher probability than the other node, respectively.

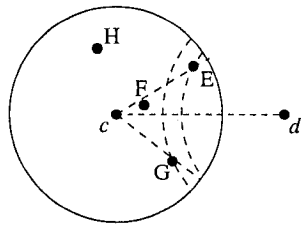

Figure 3.7: AB:GREEDY-A-DET chooses $n_1 = E$, $n_2 = G$ and AB:Compass-D-Det chooses $n_1 = F$, $n_2 = G$ as the candidate nodes Then, AB:GREEDY-A-DET chooses n_1 and AB:Compass-D-Det chooses n_2 as the next node, respectively.

Table 3.1 shows the classification of all *AB* algorithms based on how they make their initial choice of neighbors and the probability distribution to choose the final neighbor.

The algorithm *Random-Compass* proposed by Bose and Morin [BM99] in the context of triangulations is similar to an *AB* algorithm but differs in a small way. Consider the node u, a destination node d, and let $\angle xuy$ denote the angle formed by x, u, and y measured counterclockwise. In their algorithm, one neighbor chosen, $\text{ccw}(u)$, is the neighbor of u that minimizes $\angle du\{\text{ccw}(u)\}$, and the second neighbor, $\text{cw}(u)$, is the neighbor of u that minimizes $\angle\{\text{cw}(u)\}ud$. Therefore, both neighbors could lie on the same side of the $\overline{ud}$, the line passing through u and d, whereas in an *AB* algorithm if there are no neighbors on one side of the $\overline{ud}$ line, only one neighbor is considered. *Random-Compass* is shown to work with probability 1 in triangulations [BM99] and convex subdivisions [BBC$^+$02b].

Name of Algorithm	Selection of Neighbors	Probability Distribution
AB:GREEDY-U	Greedy	Uniform
AB:COMPASS-U	Compass	Uniform
AB:GREEDY-D	Greedy	Based on Distance
AB:COMPASS-D	Compass	Based on Distance
AB:COMPASS-D-DET	Compass	Deterministic
AB:GREEDY-A	Greedy	Based on Angle
AB:GREEDY-A-DET	Greedy	Deterministic
AB:COMPASS-A	Compass	Based on Angle

Table 3.1: Classification of *AB* algorithms.

3.3.4 Combined Sector and AB Approach

In this subsection we present another sub-set of algorithms which combine the sector and *AB* approaches to select the candidate nodes.

INSECTOR:AB-U: Let n_1 and n_2 be the neighbors of c, which are chosen uniformly at random from $Sector(c,d,\theta)$, from above and below the $\overline{cd}$ line, respectively. The next node x is chosen uniformly at random out of n_1 and n_2 if $Sector(c,d,\theta)$ is not empty. Otherwise, x is chosen with a probability distribution as in WEIGHTEDSECTOR from the set $N(c)$.

INSECTOR:AB-D: Let n_1 and n_2 be the neighbors of c, which are chosen uniformly at random from $Sector(c,d,\theta)$, from above and below the $\overline{cd}$ line, respectively. The next node x is chosen from n_1 and n_2 with probability $dis_2/(dis_1+dis_2)$ and $dis_1/(dis_1+dis_2)$, respectively if $Sector(c,d,\theta)$ is not empty. Otherwise, x is chosen with a probability distribution as in WEIGHTEDSECTOR from the set $N(c)$.

INSECTOR:AB-A: Let n_1 and n_2 be the neighbors of c, which are chosen uniformly at random from $Sector(c,d,\theta)$, from above and below the $\overline{cd}$ line, respectively. The next node x is chosen from n_1 and n_2 with probability $\theta_2/(\theta_1+\theta_2)$ and $\theta_1/(\theta_1+\theta_2)$, respectively if $Sector(c,d,\theta)$ is not empty. Otherwise, x is chosen with a probability distribution as in WEIGHTEDSECTOR from the set $N(c)$.

The behavior of the above mentioned algorithms is illustrated in Figure 3.8.

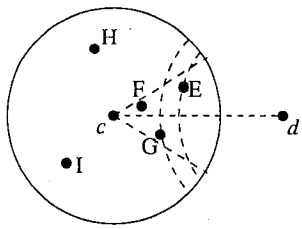

Figure 3.8: InSector:AB algorithms choose n_1 uniformly at random from E and F and $n_2 = G$ since that is the only node in the sector below the $\overline{cd}$ line. Then, INSECTOR:AB-U chooses either n_1 or n_2 with equal probability as the next node. If $n_1 = E$, then INSECTOR:AB-D chooses n_1 with higher probability than n_2 and INSECTOR:AB-A chooses n_2 with higher probability than n_1.

3.4 Properties of Randomized Routing Algorithms

In this section we summarize the properties that our randomized algorithms achieve. As stated earlier, routing algorithms in MANET must have some characteristics such as loop-freedom, distributed operation, single-path strategy, high delivery rate, low stretch factor, memorylessness, scalability, and robustness. We will show that our randomized algorithms achieve most of these characteristics successfully.

A random walk is a graph traversal technique that can visit all the vertices of a graph if it is connected, and consists of a finite number of vertices. Hence, we can consider a random walk as a randomized routing algorithm that is not defeated by any graph; however, the path taken by this technique will be much longer than the shortest path on average [BM99]. In contrast, our randomized algorithms consider a sub-set of the neighboring nodes as the candidate nodes based on different heuristics, and hence can follow a more direct path. However, sometimes these algorithms might not be able to reach the destination in a UDG.

Experimental results show that our randomized algorithms improve the packet delivery rates by overcoming the local maximum and loop. Moreover, by assigning appropriate weights to each of the candidate nodes a sub-set of these algorithms can still maintain low stretch factor. For example, in the AB algorithms we assign weights either in terms of the remaining distance from the candidate node to the destination or the angle formed between the current node, candidate node, and the destination. The latter approach outperforms all the randomized algorithms in terms of the stretch factor. Indeed, a subset of AB algorithms guarantee packet delivery for all convex subdivisions. We will now present the proof of

this claim in the following section (it is based on the same proof given by Bose *et al.* in [BBC+02b]).

Bose and Morin [BBC+02b] show that every deterministic algorithm is defeated by some convex subdivision. However, the same statement is not true for all other convex subdivisions. For instance, *Greedy* routing is not defeated by any Delaunay triangulation and *Compass* routing is not defeated by any regular triangulation. On the other hand, they show that the randomized algorithm *Random-Compass*, initially proposed in [BM99], is not defeated by any convex subdivision [BBC+02b]. A similar result holds for the compass-based randomized *AB* algorithms.

Theorem 1 *There is no convex subdivision that defeats any of the randomized compass-based AB routing algorithms,* AB:COMPASS-U, AB:COMPASS-D, *and* AB:COMPASS-A

Proof. The only situation where the *Random-Compass* algorithm in [BBC+02b] differs from the compass-based randomized *AB* algorithms in choice of neighbors is when one side of the $\overline{cd}$ line contains no neighbors of c. Assuming no three points are collinear, this can only happen when c and d are both neighbors on the convex hull, in which case the packet will be delivered directly to d. Therefore, since the probability of randomly choosing either of the neighbors does not factor in the original proof, the result follows from the proof in [BBC+02b]. □

Our experiments also show that a sub-set of our randomized algorithms that use angle to determine weight achieve a low stretch factor. To illustrate this behavior we consider the following example given by Bose and Morin in [BM99]. Here, the source and the destination are almost collinear with a third node, and hence it is possible to produce arbitrarily long thin triangles. If we consider AB:COMPASS-U for routing, the probability to follow a

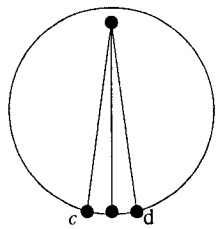

Figure 3.9: An example where AB:COMPASS-A follows a shorter path than AB:COMPASS-U.

longer path is 1/2. However, by assigning weight in terms of angle AB:COMPASS-A might take a shorter path with higher probability.

The communication and the computation costs of our randomized algorithms are also low. The communication cost at each node is $O(d)$, where d is the number of one-hop neighbors of a node. Each node needs to broadcast its coordinates to its neighbors, and also receives broadcast messages from the neighbors with their coordinates. Similarly, the computation cost at each node is also $O(d)$ since each node needs to consider only the neighboring nodes to select the next node.

Our randomized algorithms are distributed in nature, and also memoryless. An algorithm is called memoryless if no memory of previously visited nodes is necessary. Hence, all of our randomized algorithms are memoryless. Moreover, they follow a single-path strategy, that is, there is only one copy of the message available in the network. Finally, all the above mentioned characteristics of our algorithms make them suitable for a network with large number of nodes; that is, our algorithms are scalable.

Chapter 4

Performance of Randomized Routing Algorithms

In this chapter we present the simulation environment with details of simulation parameters. The performance analysis of our randomized algorithms is given next. The analysis first considers the effect of our algorithms on UDG, and then other subgraphs are taken into account. In addition, we also compare the performance of the algorithms with each other.

4.1 Simulation Environment

With the exceptions of *Greedy*, *Compass*, AB:COMPASS-D-DET, and AB:GREEDY-A-DET all of the algorithms considered in our thesis are randomized. To evaluate the performance of these algorithms we will consider their packet delivery rates and stretch factors. We first describe our simulation environment, and then the choice of algorithm-specific parameters.

We used C++ to implement our own simulator with the following setup. In the simulation experiments, a set S of n points (where $n \in \{75, 100, 125, 150\}$) is randomly generated on a square of 100m by 100m. For the transmission range of nodes, we use 15m or 18m. Experiments showed that with lower transmission radii, the graph was too often disconnected, and with higher transmission radii, the generated graphs were so dense that the delivery rate of all algorithms approached 100%. After generating $G = UDG(S)$, a source and destination node are randomly chosen. If there is no path from s to d in G, the graph is discarded, otherwise, all routing algorithms are applied in turn on G, $GG(G)$, $YG_8(G)$,

and $PLDel(G)$. Clearly, an algorithm succeeds if a path to the destination is discovered. The deterministic algorithms are deemed to fail if they enter a loop, while the randomized algorithms are considered to fail when the number of hops in the path computed so far exceeds a threshold. We will use the number of nodes, n, as this threshold (in 4.2.4, we will explore the effect of varying the threshold value). To compute the packet delivery rate, this process is repeated with 100 random graphs and the percentage of successful deliveries determined. To compute an average packet delivery rate, the packet delivery rate is determined 100 times and an average taken. Additionally, out of the 100×100 runs used to compute the average packet delivery rate, the average hop stretch factor is computed.

Several of the randomized routing algorithms use experimentally optimized parameters. In particular, FARINSECTOR depends on the parameters θ and α. A smaller value of θ clearly means a smaller number of eligible neighbors of c. Similarly, a smaller value of α means the area of the periphery is smaller compared to the area of the sector, which changes the number of eligible neighbors that are closer to the destination. We use $\theta = \pi/3$ and $\alpha = 0.6$ after comparing the performance of the algorithm with θ varying from $\pi/3$ to π and α from 0.1 to 0.9. The simulation results to determine this parameter are given in Table A.1 to Table A.4. For GREEDYINSECTOR, $\theta = \pi/3$ was found experimentally to give the best performance. Also, RANDOMINSECTOR depends on the size of the nested sectors, and the experimentally determined optimum values are $\pi/6$ and $\pi/3$, respectively. Indeed, all the weighted sector based algorithms also perform best when the size of the sector is $\pi/3$.

4.2 Discussion of Results

Detailed simulation results for all the routing algorithms, along with the associated standard deviations, are given in Tables 4.1 to Table 4.8 for the case when the transmission radius is 15 [1] for the *UDG*, the *Yao*, *Gabriel*, and *PLDel* subgraphs. In particular, we are interested in the performance of our proposed randomized routing algorithms with the previously published routing algorithms GREEDY and COMPASS.

[1]The trend of the results is the same when the transmission radius is 18, which are given in the Appendix B.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	71.23	4.12	76.43	3.93	85.96	3.56	92.40	2.13
COMPASS	72.41	4.17	78.01	3.80	87.66	3.55	93.84	2.27
BEST2COMPASS-U	81.77	3.67	85.63	3.00	92.38	2.64	96.75	1.67
BEST2GREEDY-U	79.88	3.82	83.73	3.40	90.88	2.91	95.54	2.05
FARINSECTOR	81.11	3.85	86.76	3.22	94.11	2.26	97.57	1.60
GREEDYINSECTOR	81.01	3.48	87.06	3.38	94.29	2.30	97.90	1.53
RANDOMINSECTOR	80.74	3.54	87.27	3.16	94.12	2.33	98.04	1.48
WEIGHTEDINSECTOR	80.96	3.69	87.27	3.24	94.07	2.39	97.77	1.64
WEIGHTEDSECTOR	83.23	3.50	88.35	2.99	94.60	2.22	98.03	1.43
AB:COMPASS-D-DET	72.15	4.02	77.45	3.68	87.06	3.55	93.21	2.25
AB:GREEDY-A-DET	71.82	4.16	77.30	4.00	86.81	3.45	92.90	2.02
AB:COMPASS-U	87.23	3.27	93.60	2.61	97.80	1.40	99.43	0.77
AB:GREEDY-U	87.46	2.98	92.89	2.42	97.42	1.66	99.19	0.83
AB:COMPASS-D	89.19	2.94	94.35	2.41	97.74	1.50	99.45	0.69
AB:GREEDY-D	88.42	2.94	92.93	2.41	97.56	1.63	99.24	0.77
AB:COMPASS-A	88.13	3.05	92.58	2.27	97.02	1.56	99.26	0.86
AB:GREEDY-A	86.26	3.21	90.88	2.85	95.92	1.95	98.47	1.30
INSECTOR:AB-U	83.04	3.78	88.19	2.93	94.91	2.13	98.12	1.38
INSECTOR:AB-A	82.99	3.49	88.32	3.02	94.96	2.18	98.21	1.40
INSECTOR:AB-D	83.35	3.56	88.27	3.17	94.66	2.10	98.06	1.42

Table 4.1: Average packet delivery rate and standard deviation in *UDG*, in terms of percentages, for transmission radius $r = 15$m.

4.2.1 Performance of Algorithms in *UDG*

Sector and Best-Two-Neighbors Algorithms

BEST2GREEDY-U and BEST2COMPASS-U are straightforward randomizations of the *Greedy* and *Compass* strategies, where the next node is chosen randomly from the top two candidates according to the respective heuristics. BEST2GREEDY-U is the worst of the randomized strategies in terms of delivery rate, but the best in terms of the stretch factor for both values of transmission radius. BEST2COMPASS-U has the third-best stretch factor, but second-best delivery rate among these two group of algorithms.

The sector-based algorithms improve significantly on the delivery rate of GREEDY, COMPASS, and BEST2GREEDY-U. The key idea behind the sector-based algorithms is to

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.01	0.05	1.02	0.07	1.02	0.07	1.02	0.06
COMPASS	1.05	0.11	1.08	0.13	1.09	0.14	1.10	0.14
BEST2COMPASS-U	1.89	1.50	1.69	1.22	1.47	0.86	1.34	0.63
BEST2GREEDY-U	1.82	1.37	1.61	1.11	1.41	0.88	1.28	0.55
FARINSECTOR	1.96	1.83	2.04	2.16	1.85	2.05	1.62	1.78
GREEDYINSECTOR	2.17	2.13	2.29	2.49	2.03	2.26	1.72	1.89
RANDOMINSECTOR	2.26	2.22	2.42	2.49	2.19	2.45	1.91	2.10
WEIGHTEDINSECTOR	2.24	2.14	2.40	2.52	2.20	2.45	1.90	2.08
WEIGHTEDSECTOR	1.84	1.65	1.90	1.86	1.79	1.78	1.61	1.56
AB:COMPASS-D-DET	1.03	0.09	1.05	0.10	1.06	0.11	1.06	0.11
AB:GREEDY-A-DET	1.03	0.08	1.04	0.10	1.05	0.10	1.05	0.10
AB:COMPASS-U	2.67	2.37	2.70	2.51	2.30	2.17	1.95	1.76
AB:GREEDY-U	2.61	2.36	2.55	2.39	2.21	2.12	1.87	1.82
AB:COMPASS-D	2.23	1.89	2.24	2.06	1.92	1.72	1.68	1.45
AB:GREEDY-D	2.12	1.81	2.14	2.03	1.89	1.80	1.62	1.49
AB:COMPASS-A	1.71	1.41	1.70	1.47	1.57	1.38	1.43	1.30
AB:GREEDY-A	1.68	1.52	1.68	1.62	1.55	1.62	1.39	1.38
INSECTOR:AB-U	1.84	1.61	1.90	1.79	1.80	1.80	1.62	1.57
INSECTOR:AB-A	1.82	1.64	1.92	1.87	1.81	1.87	1.63	1.55
INSECTOR:AB-D	1.84	1.61	1.93	1.80	1.81	1.83	1.62	1.47

Table 4.2: Average stretch factor and standard deviation in UDG, for transmission radius $r = 15$m.

restrict the extent to which we stray away from the direction of the destination while keeping some flexibility regarding exactly which neighbor to forward to next. The five algorithms differ in the choice of the neighbor within the sector[2]. GREEDYINSECTOR chooses the neighbor closest to the destination from within the sector, while FARINSECTOR chooses a node randomly from among the nodes closest to the destination ($Periphery(c,d,\theta,\alpha)$), and RANDOMINSECTOR chooses randomly from among the nodes closest to the direction of the destination.

The stretch factors of these three sector-based algorithms are similar, with FARINSECTOR having the best result. FARINSECTOR also allows for the choice of next neighbor from $Wing(c,d,\theta,\alpha)$, if the $Periphery(c,d,\theta,\alpha)$ is empty, on the grounds that it may be worthwhile straying outside the sector provided that we cover a lot of distance to the destination. However FARINSECTOR also is slightly more complicated to implement than the

[2]For simplicity, we use "the sector" to refer to $Sector(c,d,\theta)$ in this discussion.

other two. While the delivery rates of these three sector-based algorithms are almost identical for both values of transmission radius, FARINSECTOR appears to have a slight edge over the other two.

WEIGHTEDINSECTOR is a variation of RANDOMINSECTOR, where each node inside the sector has a positive weight in terms of the angle formed between the neighboring node, the current node c, and the destination. The next node is chosen according to a probability distribution from among the nodes inside the sector such that probability to pick a node which is closest to the $\overline{cd}$ line is high. If the sector is empty, the algorithm chooses the next node uniformly at random from the remaining neighbors of c. WEIGHTEDSECTOR differs from the WEIGHTEDINSECTOR when the $Sector(c,d,\theta)$ is empty. Instead of picking x uniformly at random from the remaining nodes in $N(c)$, it also assigns weight to those neighbors. Hence, we assign weight to all the neighbors of c to keep the next node x as close as possible to the $\overline{cd}$ line.

WEIGHTEDSECTOR has the best performance among all sector-based and *Best-Two-Neighbors* algorithms, in terms of the delivery rate with the second-best stretch factor. The RANDOMINSECTOR algorithm is worse than all other *Sector* algorithms in terms of both the delivery rate and stretch factor. Recall that RANDOMINSECTOR chooses the next node with equal probability among the candidate nodes from inside the nested sectors. Also, if the nested sectors are empty, there is a good chance of avoiding it by picking next node uniformly at random from $N(c)$, which gives a good delivery rate. However, by potentially moving far away from the direction of the destination, we may end up increasing the path length. In contrast, WEIGHTEDSECTOR still allows for moving inside $Sector(c,d,\theta)$, but by weighting the probability of the choice of each neighbor x on the angle $\angle xcd$, it reduces the chances of moving too far away from the right direction and therefore taking too long a path. As a result, WEIGHTEDSECTOR achieves significantly better stretch factor than RANDOMINSECTOR as well as a better delivery rate.

AB Algorithms

The key observation is that all the randomized *AB* algorithms have significantly higher delivery rates than *Best-Two-Neighbors* and *Sector* algorithms as well as the deterministic *AB* algorithms. The delivery rates of all the randomized *AB* algorithms are similar. Randomization keeps some flexibility in the routes, so that a dead-end on one side of $\overline{cd}$ line can always be avoided. However, the BEST2GREEDY-U, the BEST2COMPASS-U, and the

Sector algorithms may be more likely to explore the dead end and take a long time to recover from it. Therefore they may fail, while the randomized *AB* algorithms always explore both sides of the $\overline{cd}$ line with non-zero probability, which accounts for their higher delivery rates.

Interestingly, the weight assigned to the two candidate neighbors to determine the next node to forward the packet makes a difference to the stretch factor achieved by the randomized *AB* algorithms. Picking a neighbor uniformly at random from the two candidates performs most poorly, with picking a neighbor at random based on distance to the destination doing better, and picking a neighbor randomly weighted by the angles from the $\overline{cd}$ line performing best. In fact, AB:GREEDY-A and AB:COMPASS-A have the best stretch factors of all randomized algorithms. Recall that the candidate neighbors are on both sides of the $\overline{cd}$ line. In the AB:COMPASS-U and AB:GREEDY-U, the two candidates have equal probability of being chosen as the next node even if choosing one of them means moving too far from the direction of the destination or not making progress in terms of distance, thereby increasing the path length. In contrast, the other *AB* algorithms favor the node which makes more progress. For example, AB:COMPASS-A and AB:GREEDY-A still allow for moving on both sides of the $\overline{cd}$ line, but by weighting the probability of the choice of each neighbor x on the angle $\angle xcd$ (or $\angle dcx$), the chances of moving too far away from the right direction and therefore taking too long a path are reduced [FHN04].

Combined AB-Sector Algorithms

This class of algorithms brings the sector and *AB* approaches together, in an attempt to utilize both of their advantages. The algorithms apply the *AB* approach inside the specified sector. The simulation results show that the delivery rate of these algorithms is similar to that of sector-based algorithms with better stretch factor, except for the WEIGHTEDSECTOR algorithm. Moreover, if the sector is empty, instead of picking x randomly out of $N(c)$, we assign weights to those nodes. The combination of these two techniques ensures a reasonable delivery rate as well as stretch factor. However, we might still deviate from the direction of the destination since we pick the candidate nodes uniformly at random from above and below the $\overline{cd}$ line inside the sector. Hence, the delivery rate of these algorithms is less than that of the other *AB* algorithms, but higher than the *Best-Two-Neighbors* algorithms. Similarly, the stretch factor of the combined approach is between *AB* and *Sector* algorithms. Finally, the combined approach is bit more complex to implement.

Comparison of Different Classes of Algorithms

We will now present the comparison between different groups of algorithms in terms of packet delivery rate and stretch factor.

In *UDG*s, the randomized *AB* algorithms outperform all the other randomized and deterministic algorithms in terms of the packet delivery rate. The performance of the combined *AB-Sector* algorithms is the next best, but WEIGHTEDSECTOR offers the same delivery rate as this group. The performance of the remaining *Sector* and *Best-Two-Neighbors* algorithms comes next, and finally, all the deterministic algorithms have the worst delivery rates. In terms of stretch factor, the deterministic algorithms have the best stretch factors. The angle-based *AB* algorithms are the second-best in terms of the stretch factor, and the combined *AB-Sector* and *Best-Two-Neighbors* algorithms are the third-best group of algorithms. The *Sector* and distance-based *AB* algorithms come next, and WEIGHTEDSECTOR outperforms all the algorithms from these two groups. The uniform *AB* algorithms have the worst stretch factor among all the algorithms.

4.2.2 Effect of Type of Subgraph

The trend of the results in Tables B.3 to B.8 is similar within each type of graph studied. The only break with this pattern is the BEST2COMPASS-U and BEST2GREEDY-U algorithms which have equivalent performance with the *AB* algorithms in the *Gabriel* subgraph. Now we compare the performance of the algorithms between the different types of graphs in terms of delivery rates. In general, all algorithms have the best delivery rates in the *UDG* and the *PLDel* subgraph, with the *Yao* subgraph doing very slightly worse. For the *Gabriel* subgraph, the delivery rates are lower than the rest. In terms of stretch factor, all algorithms have the best stretch factor in the *UDG*, with the *PLDel* and *Yao* subgraph having slightly higher values. The highest stretch factor is achieved on the *Gabriel* subgraph. We understand the reason for these trends is the relative density of the graphs, which in turn leads to different lengths of shortest paths. As shown in Figure 4.2, the *Gabriel* subgraph has significantly lower number of edges than the *PLDel* and *Yao* subgraphs, which in turn have far fewer edges than the UDG. Figure 4.1 shows that the average length of the shortest paths in the *Gabriel* subgraph is bigger than in the *PLDel* and *Yao* subgraphs, which are in turn bigger than in the UDG. With a larger number of hops, during randomized routing, the chance of diverging from the shortest path is greater leading to longer paths during routing

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	70.24	3.99	75.04	3.80	84.52	3.98	91.47	2.39
COMPASS	70.99	4.10	76.42	4.03	85.44	3.68	92.43	2.30
BEST2COMPASS-U	82.38	3.74	86.00	3.29	92.55	2.66	96.61	1.81
BEST2GREEDY-U	80.78	4.00	84.26	3.27	91.24	2.83	95.88	2.00
FARINSECTOR	80.63	3.81	86.40	3.77	93.41	2.42	97.47	1.62
GREEDYINSECTOR	80.81	3.72	86.62	3.46	93.76	2.47	97.60	1.53
RANDOMINSECTOR	80.49	3.60	86.54	3.23	93.79	2.35	97.70	1.60
WEIGHTEDINSECTOR	80.68	3.59	86.27	3.19	93.96	2.34	97.74	1.50
WEIGHTEDSECTOR	82.78	3.45	87.60	3.15	94.07	2.32	97.62	1.50
AB:COMPASS-D-DET	70.69	4.09	75.66	3.53	85.08	3.67	92.03	2.31
AB:GREEDY-A-DET	70.74	4.27	76.11	3.92	84.68	3.70	91.76	2.36
AB:COMPASS-U	86.22	2.98	92.35	2.87	97.19	1.58	99.08	1.01
AB:GREEDY-U	85.53	3.59	91.82	2.69	96.98	1.92	99.12	0.94
AB:COMPASS-D	88.35	2.95	92.85	2.32	97.31	1.54	99.20	0.77
AB:GREEDY-D	87.28	3.26	92.49	2.82	96.79	1.75	98.94	0.87
AB:COMPASS-A	87.13	3.16	91.40	2.70	96.46	1.78	98.78	1.04
AB:GREEDY-A	86.11	3.02	90.09	2.91	95.19	2.06	98.38	1.52
INSECTOR:AB-U	82.81	3.43	87.63	3.05	94.18	2.30	97.69	1.59
INSECTOR:AB-A	82.81	3.60	87.68	3.04	93.98	2.42	97.55	1.53
INSECTOR:AB-D	82.73	3.68	87.49	3.17	94.14	2.45	97.82	1.50

Table 4.3: Average packet delivery rate and standard deviation in *Yao* graph, in terms of percentages, for transmission radius $r = 15$m.

on average. With the longer paths, the chance of terminating a path at the threshold is also greater leading to lower delivery rates. In general, we can conclude that the performance of the algorithms on both the *Yao* and *PLDel* subgraphs is comparable to that in the original unit disk graph, while the subgraphs have far fewer edges to maintain than the UDG [FHN04].

4.2.3 Effect of Number of Nodes

In this section we describe the effect of number of nodes on the performance of the algorithms.

In general, as the number of nodes grows, the delivery rate increases in all the graphs studied. In terms of average stretch factor, however, the results are different for the five classes of algorithms studied.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	66.26	4.35	70.01	4.43	78.88	3.99	86.13	3.28
COMPASS	66.17	4.68	70.03	4.37	78.62	4.03	85.76	3.34
BEST2COMPASS-U	78.75	4.06	85.40	3.63	92.48	2.51	96.50	1.88
BEST2GREEDY-U	78.28	3.57	85.37	3.40	92.00	2.84	96.23	1.85
FARINSECTOR	77.12	3.50	82.36	4.17	90.35	2.77	95.11	2.03
GREEDYINSECTOR	76.60	3.78	82.78	4.18	90.19	3.10	95.28	1.96
RANDOMINSECTOR	76.52	3.76	83.05	3.55	90.56	2.84	95.31	2.03
WEIGHTEDINSECTOR	76.84	3.61	82.71	3.65	90.49	2.86	95.27	1.82
WEIGHTEDSECTOR	78.94	3.67	84.22	3.57	91.12	2.72	95.40	1.79
AB:COMPASS-D-DET	66.25	4.36	70.08	4.42	78.92	3.93	86.07	3.28
AB:GREEDY-A-DET	66.14	4.65	69.96	4.41	78.54	4.10	85.80	3.28
AB:COMPASS-U	80.23	3.67	87.07	3.70	93.65	2.14	97.54	1.46
AB:GREEDY-U	80.25	3.85	87.15	3.81	93.62	2.50	97.32	1.64
AB:COMPASS-D	83.74	3.24	88.76	3.64	94.84	2.27	97.99	1.56
AB:GREEDY-D	83.10	3.63	88.58	3.42	94.71	2.32	98.10	1.37
AB:COMPASS-A	82.81	3.30	87.96	3.47	94.00	2.35	97.29	1.57
AB:GREEDY-A	83.30	3.09	87.66	3.14	93.67	2.30	97.28	1.66
INSECTOR:AB-U	78.81	3.63	84.17	3.82	91.21	2.68	95.56	2.20
INSECTOR:AB-A	79.15	3.54	84.25	3.80	91.30	2.72	95.63	1.87
INSECTOR:AB-D	78.86	3.38	83.97	3.74	91.19	2.47	95.74	1.94

Table 4.4: Average packet delivery rate and standard deviation in *Gabriel* graph, in terms of percentages, for transmission radius $r = 15$m.

For the deterministic algorithms, the stretch factors increase very slightly as the number of nodes increases. For the straightforward randomizations, BEST2COMPASS-U and BEST2GREEDY-U, as the number of nodes increases, the average stretch factors decrease on the *UDG* and *Yao* subgraph. However, this value first increases with the increasing number of nodes, and then starts decreasing on *PLDel* and *Gabriel* subgraphs.

The sector-based and combined *AB-Sector* algorithms show another behavior, the average stretch factors first increase and then start to decrease with the increasing number of nodes for all the graphs. Finally, the randomized AB algorithms show yet another trend. The stretch factors in the *UDG* decrease as the number of nodes increases. For the subgraphs of the *UDG*, the average stretch factors initially increase, and then the values decrease.

For larger values of n, the number of possible paths available to the randomized algorithms increase, so the stretch factor can be expected to decrease. For instance, in a denser

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	70.72	4.89	75.04	4.37	84.73	3.69	92.71	2.61
COMPASS	71.40	4.77	76.25	3.98	85.64	3.62	93.25	2.54
BEST2COMPASS-U	81.91	4.13	85.37	3.46	92.27	2.96	96.96	1.74
BEST2GREEDY-U	80.39	4.01	83.78	3.82	90.62	3.18	96.12	1.88
FARINSECTOR	80.90	4.39	85.76	3.40	93.54	2.51	97.96	1.39
GREEDYINSECTOR	81.13	4.01	86.38	3.52	93.96	2.61	98.32	1.14
RANDOMINSECTOR	80.78	4.34	86.38	3.63	94.11	2.41	98.17	1.22
WEIGHTEDINSECTOR	81.34	3.97	86.54	3.43	94.03	2.35	98.23	1.21
WEIGHTEDSECTOR	82.99	4.21	87.57	3.40	94.17	2.55	98.23	1.17
AB:COMPASS-D-DET	71.06	4.88	75.55	4.24	85.36	3.59	92.99	2.48
AB:GREEDY-A-DET	71.13	4.77	75.79	4.09	85.30	3.56	93.21	2.48
AB:COMPASS-U	86.36	3.44	92.40	2.86	97.47	1.47	99.47	0.78
AB:GREEDY-U	86.26	4.12	92.08	2.78	97.23	1.66	99.17	0.89
AB:COMPASS-D	88.58	3.06	93.14	2.57	97.64	1.58	99.45	0.85
AB:GREEDY-D	87.67	3.76	92.46	2.74	97.21	1.64	99.28	0.88
AB:COMPASS-A	87.62	3.59	91.59	2.61	96.80	1.85	99.25	0.87
AB:GREEDY-A	86.21	3.89	89.62	3.22	95.63	2.21	98.72	1.10
INSECTOR:AB-U	82.87	3.85	87.36	3.75	94.52	2.56	98.27	1.23
INSECTOR:AB-A	83.02	3.91	87.52	3.09	94.19	2.31	98.27	1.33
INSECTOR:AB-D	83.17	3.74	87.45	3.30	94.34	2.50	98.08	1.24

Table 4.5: Average packet delivery rate and standard deviation in *PLDel* graph, in terms of percentages, for transmission radius $r = 15$m.

graph, the algorithm can recover from a bad path earlier, which leads to a lower stretch factor for the algorithm. Also, with the increased density of nodes, the behavior of the straightforward randomizations should approach that of the corresponding *AB* randomized algorithm with uniform weighting, which is also supported by the data. Finally, the randomized *AB* algorithms can be seen as a cross between the deterministic algorithms and the straightforward randomized algorithms. At lower values of n, since there are fewer choices of neighbors, their performance is closer to that of the deterministic algorithms.

4.2.4 Effect of Threshold

This section focuses on the behavior of our randomized algorithms with the variation in the threshold value, where we consider only *UDG*s.

Figure 4.3 and 4.4 show the effect of varying the threshold value (which was set to the

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.08	0.14	1.11	0.16	1.15	0.18	1.18	0.20
COMPASS	1.11	0.16	1.16	0.19	1.20	0.21	1.25	0.23
BEST2COMPASS-U	2.17	1.72	2.05	1.58	1.86	1.34	1.74	1.13
BEST2GREEDY-U	2.16	1.83	2.00	1.52	1.80	1.24	1.68	0.97
FARINSECTOR	2.18	2.11	2.30	2.29	2.21	2.34	2.09	2.07
GREEDYINSECTOR	2.35	2.27	2.48	2.45	2.38	2.42	2.21	2.33
RANDOMINSECTOR	2.42	2.36	2.58	2.58	2.50	2.69	2.31	2.46
WEIGHTEDINSECTOR	2.40	2.29	2.58	2.58	2.47	2.56	2.28	2.39
WEIGHTEDSECTOR	1.94	1.75	2.03	1.83	1.97	1.90	1.88	1.89
AB:COMPASS-D-DET	1.09	0.15	1.13	0.17	1.17	0.19	1.20	0.21
AB:GREEDY-A-DET	1.09	0.15	1.14	0.18	1.18	0.20	1.22	0.22
AB:COMPASS-U	2.87	2.55	2.99	2.70	2.72	2.55	2.41	2.26
AB:GREEDY-U	2.85	2.55	2.98	2.75	2.70	2.69	2.33	2.13
AB:COMPASS-D	2.41	2.08	2.50	2.24	2.27	2.08	2.02	1.70
AB:GREEDY-D	2.33	1.97	2.41	2.09	2.26	2.12	2.01	1.76
AB:COMPASS-A	1.86	1.57	1.89	1.71	1.77	1.62	1.66	1.41
AB:GREEDY-A	1.81	1.59	1.88	1.80	1.76	1.68	1.66	1.58
INSECTOR:AB-U	1.92	1.71	2.04	1.88	2.00	1.94	1.87	1.72
INSECTOR:AB-A	1.93	1.74	2.03	1.89	1.97	1.93	1.85	1.65
INSECTOR:AB-D	1.93	1.67	2.07	2.00	2.01	1.95	1.90	1.80

Table 4.6: Average stretch factor and standard deviation in *Yao* graph, for transmission radius $r = 15$m.

number of nodes in the simulations in Section 4.2) on the performance of the randomized algorithms. We consider one representative algorithm from each group of randomized algorithms. The results for other algorithms are similar.

It is interesting to note that the average delivery rate of BEST2COMPASS-U is almost same as the uniform randomized AB algorithms when the threshold is $n/2$. The same statement is true for WEIGHTEDSECTOR and INSECTOR:AB-A algorithms, and the delivery rate increases slowly with the increasing threshold value. As the threshold increases, the average delivery rate of BEST2COMPASS-U increases only slightly whereas the AB algorithms much improve their average delivery rates. However, as the threshold reaches $2n$, AB:COMPASS-A starts falling behind. We expect that a similar pattern would hold for the subgraphs of the *UDG*.

For the stretch factors shown in Figure 4.4, for small thresholds of $n/2$, the AB:COMPAS S-A algorithm has the smallest stretch factors, followed by the WEIGHTEDSECTOR and

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.23	0.26	1.34	0.30	1.42	0.33	1.52	0.36
COMPASS	1.24	0.26	1.35	0.30	1.44	0.34	1.55	0.38
BEST2COMPASS-U	3.49	2.97	3.82	3.22	3.74	3.22	3.63	3.06
BEST2GREEDY-U	3.49	2.95	3.89	3.39	3.77	3.18	3.67	3.03
FARINSECTOR	2.72	2.47	3.10	2.82	3.33	3.15	3.38	3.01
GREEDYINSECTOR	2.84	2.61	3.29	2.91	3.52	3.42	3.52	3.29
RANDOMINSECTOR	2.80	2.53	3.33	3.07	3.48	3.29	3.51	3.14
WEIGHTEDINSECTOR	2.83	2.62	3.29	3.05	3.52	3.22	3.53	3.30
WEIGHTEDSECTOR	2.17	1.83	2.43	2.16	2.54	2.35	2.51	2.12
AB:COMPASS-D-DET	1.23	0.26	1.34	0.30	1.42	0.33	1.52	0.36
AB:GREEDY-A-DET	1.24	0.26	1.35	0.30	1.44	0.34	1.54	0.38
AB:COMPASS-U	3.52	3.09	4.18	3.64	4.25	3.80	4.15	3.67
AB:GREEDY-U	3.54	3.08	4.12	3.53	4.28	3.79	4.18	3.72
AB:COMPASS-D	2.97	2.48	3.41	2.91	3.49	3.02	3.47	2.92
AB:GREEDY-D	2.95	2.49	3.45	2.93	3.51	3.07	3.46	2.97
AB:COMPASS-A	2.20	1.91	2.44	2.29	2.46	2.29	2.41	2.18
AB:GREEDY-A	2.18	1.86	2.39	2.13	2.44	2.21	2.40	2.18
INSECTOR:AB-U	2.18	1.88	2.45	2.24	2.55	2.35	2.51	2.13
INSECTOR:AB-A	2.16	1.85	2.44	2.11	2.52	2.25	2.53	2.16
INSECTOR:AB-D	2.14	1.79	2.38	2.00	2.53	2.23	2.55	2.26

Table 4.7: Average stretch factor and standard deviation in *Gabriel* graph, for transmission radius $r = 15$m.

INSECTOR:AB-A algorithms. The third-best algorithm is BEST2COMPASS-U, then the AB:COMPASS-D algorithm, with the AB:COMPASS-U algorithm having the largest average stretch factors. As the threshold increases, the average stretch factor of the BEST2COMPASS-U algorithm grows most slowly and the relative values of the WEIGHTEDSECTOR, INSECTOR:AB-A, and AB algorithms maintain the same ordering. By a threshold of $7n/4$, the average stretch factor of the BEST2COMPASS-U is the lowest.

From the information in Figure 4.3, the best balance in terms of maximizing the average delivery rate while maintaining a low average stretch factor is obtained for threshold values between n and $3n/2$. Thus our choice of n for a threshold is appropriate.

We have also calculated the stretch factor based on Euclidean distance, but the results are similar to that of the hop-based measure (which is not surprising since over many random paths for a fixed transmission range, the edge used for each hop would have an average length).

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.09	0.15	4.37	1.14	1.20	0.20	1.25	0.22
COMPASS	1.12	0.17	3.98	1.18	1.24	0.22	1.30	0.25
BEST2COMPASS-U	2.25	1.80	3.46	2.12	1.92	1.24	1.81	1.03
BEST2GREEDY-U	2.18	1.71	3.82	2.06	1.91	1.31	1.80	1.06
FARINSECTOR	2.22	2.05	3.40	2.33	2.33	2.38	2.14	1.96
GREEDYINSECTOR	2.33	2.13	3.52	2.55	2.43	2.39	2.25	2.21
RANDOMINSECTOR	2.40	2.19	3.63	2.61	2.50	2.50	2.31	2.22
WEIGHTEDINSECTOR	2.41	2.25	3.43	2.61	2.51	2.56	2.27	2.02
WEIGHTEDSECTOR	1.91	1.61	3.40	2.07	1.97	1.77	1.87	1.64
AB:COMPASS-D-DET	1.10	0.15	4.24	1.15	1.20	0.20	1.25	0.23
AB:GREEDY-A-DET	1.11	0.16	4.09	1.17	1.23	0.22	1.28	0.24
AB:COMPASS-U	2.89	2.62	2.86	3.00	2.70	2.46	2.38	2.16
AB:GREEDY-U	2.88	2.58	2.78	3.01	2.73	2.62	2.39	2.12
AB:COMPASS-D	2.38	2.00	2.57	2.49	2.29	1.98	2.02	1.55
AB:GREEDY-D	2.32	1.91	2.74	2.49	2.32	2.10	2.05	1.72
AB:COMPASS-A	1.80	1.45	2.61	1.90	1.81	1.61	1.65	1.21
AB:GREEDY-A	1.82	1.61	3.22	1.90	1.78	1.59	1.66	1.30
INSECTOR:AB-U	1.91	1.61	3.75	2.05	1.99	1.82	1.87	1.55
INSECTOR:AB-A	1.93	1.71	3.09	2.07	2.01	1.91	1.84	1.49
INSECTOR:AB-D	1.92	1.61	3.30	2.06	2.03	1.92	1.84	1.47

Table 4.8: Average stretch factor and standard deviation in *PLDel* graph, for transmission radius $r = 15$m.

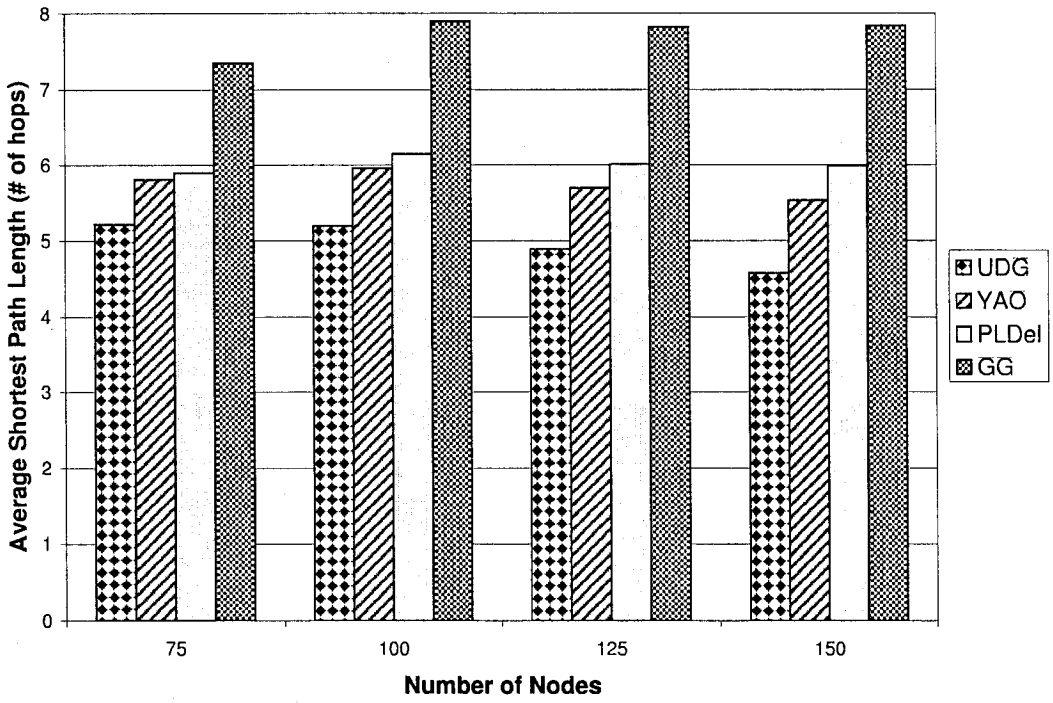

Figure 4.1: Average shortest path lengths in *UDG* and subgraphs for different values of n. Here, $r = 15$m.

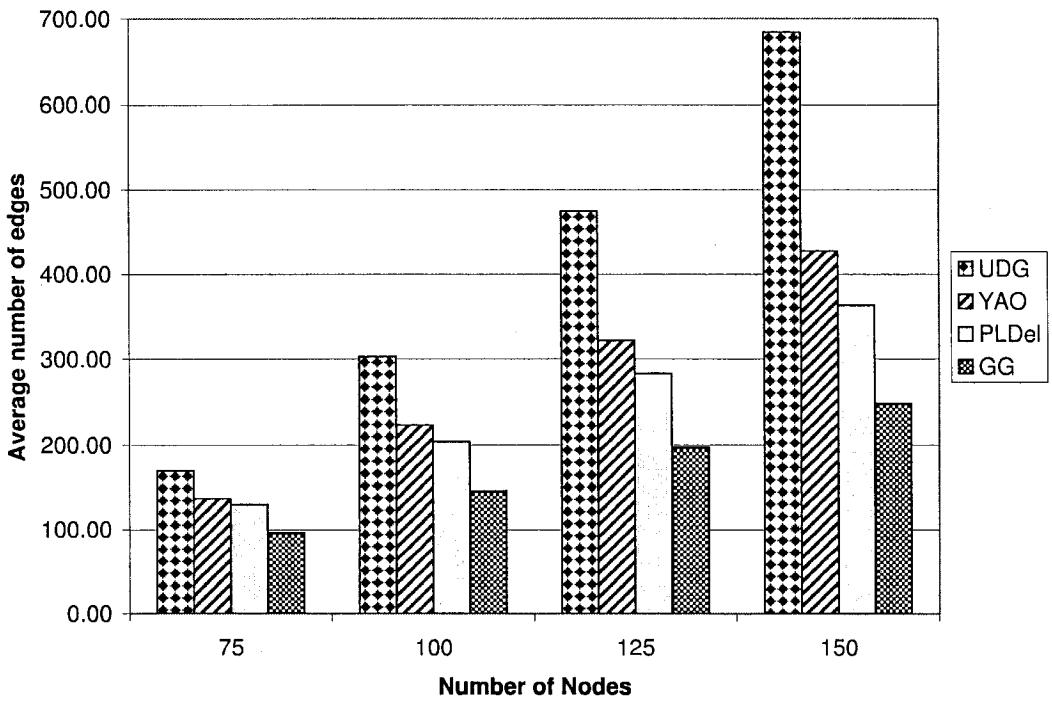

Figure 4.2: Average number of edges in *UDG* and subgraphs for different values of n. Here, $r = 15$m.

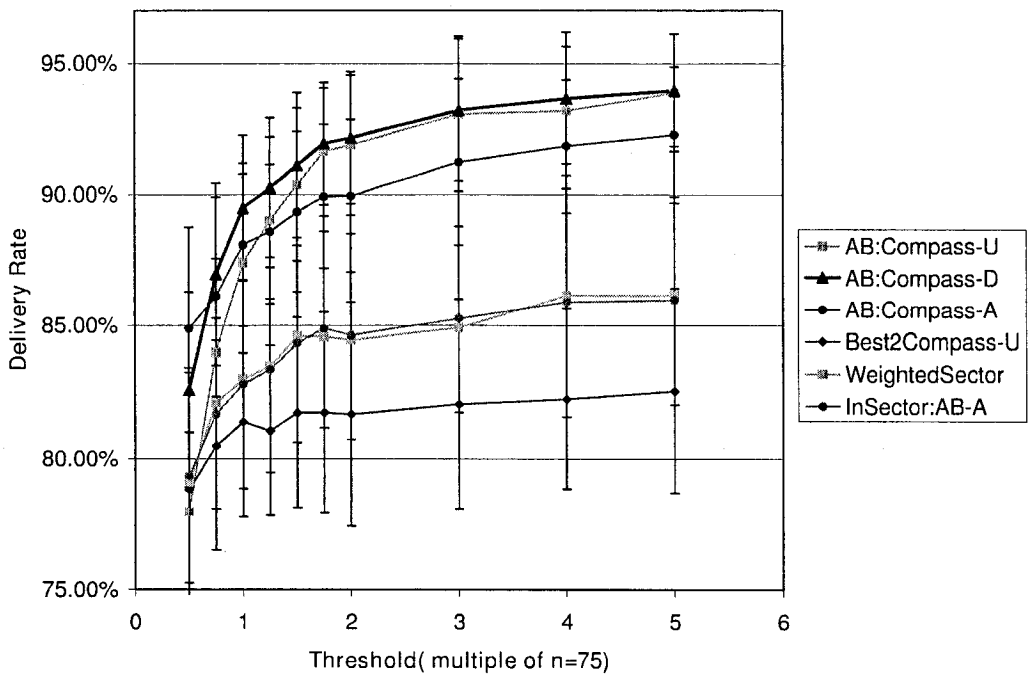

Figure 4.3: The effect of changing the threshold on the average packet delivery rates for $n = 75$ and $r = 15$m. The error bars indicate the standard deviation.

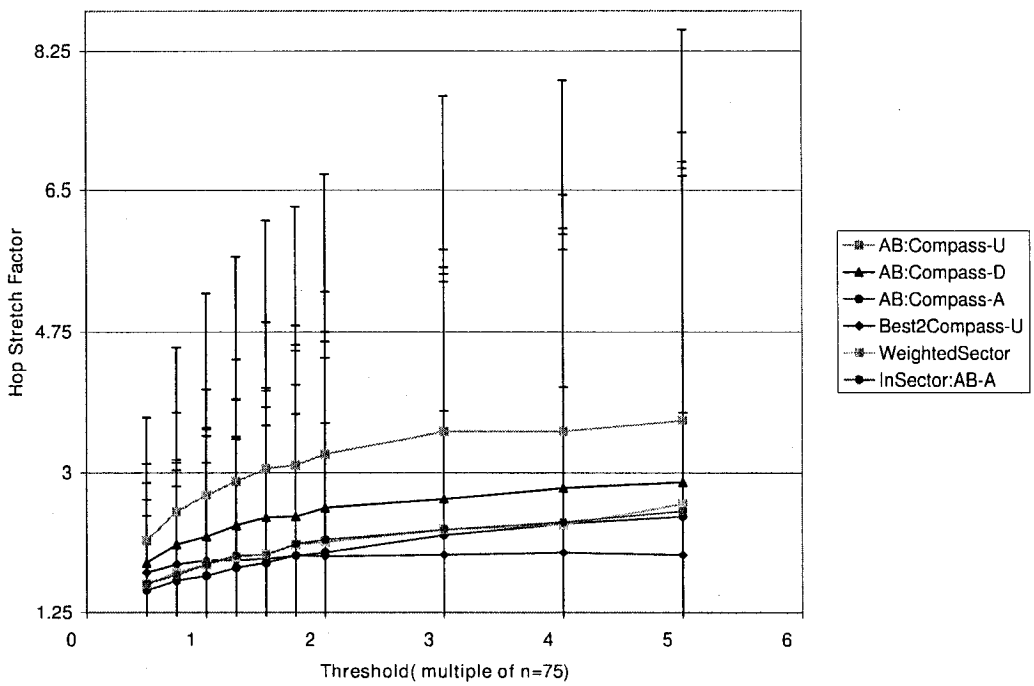

Figure 4.4: The effect of changing the threshold used on the average hop stretch factors for $n = 75$ and $r = 15$m. The error bars indicate the standard deviation.

Chapter 5

Discussion and Open Problems

In this chapter we summarize the results of this thesis, and then define several interesting problems left open by this thesis.

5.1 Conclusions

In this thesis we presented several randomized position-based algorithms for routing, categorized as *Sector*, *Best-Two-Neighbors*, and *AB* algorithms. Essentially, to determine the next node at any point, the *Best-Two-Neighbors* algorithms consider the top two candidates according to a specific heuristic; the *Sector* algorithms confine the choice of neighbors to the defined sector; and the *AB* algorithms pick one candidate above and one below the line between the current node to the destination, by using a heuristic such as greedy or directional routing, and then pick between these two candidates by flipping a biased coin.

It is immediately evident from the simulation results that all the deterministic algorithms have the worst delivery rates but the best stretch factors. All the randomized algorithms improve on the delivery rates of the two deterministic algorithms in terms of delivery rate. This is simply because the randomized algorithms are able to continue and find alternative and possibly longer routes even when encountering the same node again in a path. However, this also means that the extra paths found can be quite long, and this contributes to the higher stretch factors of the randomized algorithms. In particular, *AB* algorithms outperform the other randomized algorithms. However, in terms of average hop count stretch factors, the deterministic algorithms outperform all the randomized

algorithms, with GREEDY having the best stretch factor. The fact that the randomized algorithms are able to continue and find alternative and possibly longer routes even when encountering the same node again in a path accounts for both the higher delivery rates and the higher stretch factors as compared to the deterministic algorithms. By using weighted randomization based on the angles created by the candidate neighbors and $\overline{cd}$ line we can maintain the improved delivery rates while greatly reducing the average stretch factors of the randomized algorithms. We also presented a combined approach of *Sector* and *AB* algorithms, but their performance is not as impressive as the *AB* algorithms.

5.2 Open Problems

We perform the routing under the circumstance of static nodes. However, in reality nodes move unpredictably and randomly. The movement of nodes during routing and topology construction might create a temporary loop. In [GSB03] the authors refer to this kind of loop as a mobility-caused loop. This temporary loop might occur because of the sudden appearance of a node in the neighborhood of another node during the routing. It is not easy to detect this kind of situation since it might occur on-the-fly. Hence, constructing a localized routing scheme which can cope with node mobility is a challenging open problem.

One obvious question regarding our randomized algorithms is whether it is possible to combine these algorithms with the algorithms that guarantee the message delivery. We have already described the effect of the threshold on the message delivery. The key observation is that if we increase the threshold , then after a certain point the delivery rate does not increase much, whereas the stretch factor starts to increase significantly. Instead of increasing the threshold, we can try to identify the optimal value of the threshold where the algorithm can have a high chance of delivering the packets. Then, instead of dropping the packet when the threshold is reached , our algorithms may switch to the *Face* routing mode. The packet will be routed through the faces of the underlying planar subgraph and switched back to the random mode at some point. Now, the next question is when the algorithm should switch back to the randomized routing mode. The idea behind this combined approach is to utilize the benefit of the two approaches to ensure guaranteed packet delivery. It would be interesting to observe the behavior of the randomized algorithms under this new circumstance.

Another observation about the underlying topology is as follows. Bose and Morin

[BM99] proved that *Greedy*, *Compass*, and *Random Compass* guarantee packet delivery when the network topology is a Delaunay triangulation. However, the same statement is not true for *PLDel* since each of its faces is not necessarily a triangle. Hence, the packet might deviate from the direction of the destination. Moreover, the construction of *PLDel* assumes that the nodes are static or can be viewed as static during a reasonable period of time. This is acceptable in sensor networks, but not in a situation where nodes move frequently and unpredictably. Based on these observations, we are interested in constructing and maintaining the triangulation in a distributed manner such that node mobility is allowed. Indeed, it should still maintain all the attractive properties of *PLDel*.

Also, we have proved that a sub-set of *AB* algorithms deliver packets with probability 1 in convex subdivisions. It would be interesting to find out if all randomized *AB* algorithms can deliver packets with probability 1 in convex subdivisions. In general, we are interested in characterizing the kinds of networks for which our randomized algorithms have a delivery rate of 100%.

Finally, we have seen that in [BFNO01] the authors consider irregular transmission range of the mobile nodes, which might occur due to obstacles. The work is unique in the MANET literature, and further investigation of the behavior of randomized algorithms under such a varying radio range model would be an interesting area for future research. Also, Boukerche and Das in [BD03] proposed a randomized version of DSDV algorithm called R-DSDV. The algorithm reduces message congestion due to packet forwarding, and hence the delay. Quality-Of-Service(QoS) routing considers these parameters as part of the performance analysis. Enhancing our algorithms to consider quality of service issues is another interesting area of research.

Bibliography

[ALW+03] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[BBC+01] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.P. Hubaux, and J. Y.Le Boudec. Self-organization in mobile ad hoc networks: the approach of terminodes. Technical Report IEEE Communication Magazine, IEEE, June 2001.

[BBC+02a] P. Bose, A. Brodnik, S. Carlsson, E.D. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J.I. Munro. Online routing in convex subdivisions. *International Journal of Computational Geometry*, 12(4):283–295, 2002.

[BBC+02b] P. Bose, A. Brodnik, S. Carlsson, E.D. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J.I. Munro. Online routing in convex subdivisions. *International Journal of Computational Geometry*, 12(4):283–295, 2002.

[BCSW98] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *4th ACM/IEEE Conference on Mobile Computing and Networking (Mobicom '98)*, pages 76–84, 1998.

[BD03] A. Boukerche and S.K. Das. Congestion control performance of R-DSDV protocol in multihop wireless ad hoc networks. *ACM/Kluwer Journal on Wireless Networks*, 9(3):261–270, March 2003.

[BDEK02] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and betaskeletons. In *Proceedings of the Latin American Theoretical Infocomatics (LATIN)*, pages 479–493, 2002.

[BFNO01] L. Barriere, P. Fraignaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges.

In *Proc. of 5th ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2001.

[BGM04] P. Bose, J. Gudmundsson, and P. Morin. Ordered theta graphs. *Computational Geometry: Theory and Applications*, 28(1):11–18, 2004.

[BM99] P. Bose and P. Morin. Online routing in triangulations. In *10th Annual International Symposium on Algorithms and Computation (ISAAC '99)*, pages 113–122, 1999.

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *ACM/IEEE Conference on Mobile Computing and Networking (Mobicom '98)*, pages 85–97, 1998.

[BMSJ99] P. Bose, P. Morin, I. Stojmenovic, and J.Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *3rd Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, 1999.

[BOT01] B. Bellur, R. G. Ogier, and F. L. Templin. Topology broadcast based on reverse-path forwarding (tbrpf). Technical Report Internet Draft, draft-ietf-manet-tbrpf-01.txt, IETF, June 2001.

[CCL03] I. Chlamtac, M. Conti, and J. Liu. Mobile ad hoc networking: Imperatives and challenges. *Ad Hoc Network Journal*, 1(1):13–64, July 2003.

[Che86] L. P. Chew. There is a planar graph almost as good as the complete graphs. In *2nd Annual ACM Symposium on Computational Geometry*, pages 169–177, 1986.

[Che89] L. P. Chew. There are planar graphs almost as good as the complete graphs. *Journal of Computer and Systems Sciences*, 39(2):205–219, 1989.

[CL85] I. Chlamtac and A. Lerner. Link allocation in mobile radio networks with noise channel. In *IEEE Globecom*, December 1985.

[CL87] I. Chlamtac and A. Lerner. Fair algorithms for maximal link activation in multi-hop radio networks. *IEEE Transactions on Communications*, 35(7):739–746, 1987.

[DFS90] D. P. Dobkin, S.J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as the complete graphs. *Discrete and Computational Geometry*, 5:399–407, 1990.

[DRWT97] R. Dube, C. D. Rais, K. Wang, and S. K. Tripathi. Signal stability based adaptive routing (ssr alt ssa) for ad hoc mobile networks. *IEEE Personal Communications*, February 1997.

[FHN04] T. Fevens, I.T. Haque, and L. Narayanan. A class of randomized routing algorithms in mobile ad hoc networks. In *AlgorithmS for Wireless and mobile Networks (A_SWAN 2004)*, August 2004.

[Fin87] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, USC ISI, Marina del Ray, CA, March 1987.

[FL01] J. Freebersyser and B. Leiner. A dod perspective on mobile ad hoc networks. In C.E. Perkins and C. Perkins, editors, *Ad Hoc Networking*, pages 29–51. Addison-Wesley, 2001.

[GGH+01] J. Gao, L.J. Guibas, J. Hershburger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Second ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, pages 45–55, 2001.

[GS69] K. Gabriel and R. Sokal. A new statistical approach to geoghaphic variation analysis. In *Systematic Zoology*, pages 259–278, 1969.

[GSB03] S. Giordano, I. Stojmenovic, and Lj. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In X. Cheng, X. Huang, and D.Z. Du, editors, *Ad Hoc Wireless Networking*. Kluwer, December 2003.

[HL86] T. C. Hou and V.O.K. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[HP97] Z. J. Haas and M. R. Pearlman. The zone routing protocol(zrp) for ad hoc networks. Technical Report Internet Draft, draft-haas-zone-routing-protocol-00.txt, IETF, November 1997.

[JMHJ02] D. Johnson, D. Maltz, Y-C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (dsr). Technical Report Internet Draft, draft-ietf-manet-dsr-07.txt (work in progress), IETF, February 2002.

[JMQ+03] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized link state routing protocol. Technical Report Internet Draft,draft-ietf-manet-olsr-11.txt, IETF, July 2003.

[JNL99] M. Joa-Ng and I. Lu. A peer-to-peer zone-based two-level link state routing for ad hoc wireless networks. *IEEE Selected Areas in Communications*, 17(8):1415–1425, 1999.

[KG89] J. M. Keil and C. A. Gutwin. The delaunay triangulation closely approximates the complete euclidean graph. In *1st Workshop on Algorithms and Data Structures*, pages 47–56, 1989.

[KG92] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992.

[KK00] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. of 6th ACM Conference on Mobile Computing and Networking (Mobicom '00)*, 2000.

[KSU99] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Canadian Conference on Computational Geometry (CCCG '99)*, pages 51–54, 1999.

[KV98] Y.B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *4th ACM/IEEE Conference on Mobile Computing and Networking (Mobicom '98)*, pages 66–75, 1998.

[LCW02] X. Y. Li, G. Calinescu, and P. J. Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *21st Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM)*, 2002.

[Li03a] X.Y. Li. Applications of computational geometry in wireless ad hoc networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Ad Hoc Wireless Networking*. IEEE Press, 2003.

[Li03b] X.Y. Li. Topology control in wireless ad hoc networks. In X. Cheng, X. Huang, and D. Z. Du, editors, *Ad Hoc Networking*. Kluwer, 2003.

[LLS01] X. Lin, M. Lakshdisi, and I. Stojmenovic. Location based localized alternate, disjoint, multi-path and component routing algorithms for wireless networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 287–290, USA, October 2001.

[LS03] X. Lin and I. Stojmenovic. Location based localized alternate, disjoint and multi-path routing algorithms for wireless networks. *Journal of Parallel and Distributed Computing*, 63(1):22–32, 2003.

[LTS01] W.H. Liao, Y.C. Tseng, and J.P. Sheu. GRID: A fully location-aware routing protocols for mobile ad hoc networks. *Telecomm. Systems*, 18(1):37–60, 2001.

[LWW01] X.-Y. Li, P.-J. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *IEEE Int. Conf. on Computer Communications and Networks (ICCCN01)*, pages 564–567, 2001.

[MC98] J. P. Macker and M. S. Corson. Mobile ad hoc networking and the ietf. *Mobile Computing and Communications Review*, 2(1):9–14, 1998.

[MGLA96] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. In *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, pages 183–197, October 1996.

[PB94] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[PC01] V. Park and S. Corson. Temporally-ordered routing algorithm (tora). Technical Report Internet Draft, draft-ietf-manet-tora-spec- 03.txt, work in progress, IETF, June 2001.

[Per01] C. E. Perkins. Ad hoc networking an introduction. In C.E. Perkins and C. Perkins, editors, *Ad Hoc Networking*, pages 1–28. Addison-Wesley, 2001.

[PR99] C.E. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[PT96] C. E. Phillips and P. J. Taylor. *Theory and Applications of Numerical Analysis*. Academic Press, 1996.

[Raj02] R. Rajaraman. Topology control and routing in ad hoc networks: A survey. *ACM SIGACT News*, 33(2):60–73, 2002.

[Roy03] E. Royer. Routing approaches in mobile ad hoc networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Mobile Ad Hoc Networking*. IEEE Press Wiley, 2003.

[RR02] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: Challenges and directions. *IEEE Communications Magazine*, 40(5):20–22, May 2002.

[RT99] E. Royer and C. K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, 1999.

[SL01] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[TK84] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[Toh97] C-K. Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communications*, 4(2):1–36, 1997.

[Tou80] G. Toussiant. The relative neighborhood graph of finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[Yao82] A.C.C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Computing*, 11(0):721–736, 1982.

Appendix A

Parameter Selection

We present the simulation results for choosing parameters for our randomized algorithm. The following tables show why we chose $\pi/3$ as our sector size in most cases.

α	$Sector(c,d,\frac{\pi}{3})$	$Sector(c,d,\frac{\pi}{2})$	$Sector(c,d,\frac{2\pi}{3})$	$Sector(c,d,\frac{5\pi}{6})$	$Sector(c,d,\pi)$
0.1	81.00	80.00	79.00	80.00	80.00
0.2	81.00	79.00	80.00	80.00	80.00
0.3	81.00	80.00	80.00	79.00	79.00
0.4	81.00	80.00	79.00	79.00	79.00
0.5	81.00	79.00	79.00	78.00	78.00
0.6	82.00	79.00	79.00	78.00	78.00
0.7	81.00	80.00	79.00	78.00	77.00
0.8	80.00	79.00	79.00	78.00	79.00
0.9	80.00	79.00	79.00	79.00	79.00

Table A.1: Average packet delivery rate of FARINSECTOR for different values of π and α, in terms of percentages, for transmission radius $r = 15$m and $n = 75$.

α	$Sector(c,d,\frac{\pi}{3})$	$Sector(c,d,\frac{\pi}{2})$	$Sector(c,d,\frac{2\pi}{3})$	$Sector(c,d,\frac{5\pi}{6})$	$Sector(c,d,\pi)$
0.1	87.00	85.00	84.00	84.00	84.00
0.2	87.00	85.00	84.00	83.00	84.00
0.3	87.00	85.00	83.00	83.00	84.00
0.4	87.00	84.00	83.00	83.00	83.00
0.5	86.00	84.00	83.00	82.00	82.00
0.6	87.00	85.00	83.00	82.00	83.00
0.7	86.00	85.00	83.00	83.00	82.00
0.8	85.00	84.00	83.00	83.00	83.00
0.9	85.00	83.00	83.00	83.00	83.00

Table A.2: Average packet delivery rate of FARINSECTOR for different values of π and α, in terms of percentages, for transmission radius $r = 15$m and $n = 100$.

α	$Sector(c,d,\frac{\pi}{3})$	$Sector(c,d,\frac{\pi}{2})$	$Sector(c,d,\frac{2\pi}{3})$	$Sector(c,d,\frac{5\pi}{6})$	$Sector(c,d,\pi)$
0.1	94.00	93.00	92.00	92.00	93.00
0.2	94.00	93.00	92.00	92.00	93.00
0.3	94.00	93.00	92.00	92.00	93.00
0.4	94.00	93.00	92.00	92.00	93.00
0.5	94.00	93.00	92.00	91.00	92.00
0.6	94.00	93.00	92.00	92.00	92.00
0.7	93.00	93.00	92.00	91.00	91.00
0.8	93.00	93.00	92.00	91.00	92.00
0.9	93.00	92.00	92.00	92.00	92.00

Table A.3: Average packet delivery rate of FARINSECTOR for different values of π and α, in terms of percentages, for transmission radius $r = 15$m and $n = 125$.

α	$Sector(c,d,\frac{\pi}{3})$	$Sector(c,d,\frac{\pi}{2})$	$Sector(c,d,\frac{2\pi}{3})$	$Sector(c,d,\frac{5\pi}{6})$	$Sector(c,d,\pi)$
0.1	98.00	97.00	97.00	97.00	97.00
0.2	98.00	97.00	96.00	97.00	97.00
0.3	98.00	97.00	96.00	96.00	97.00
0.4	98.00	97.00	96.00	96.00	96.00
0.5	98.00	97.00	96.00	96.00	97.00
0.6	98.00	97.00	96.00	96.00	97.00
0.7	97.00	97.00	96.00	96.00	96.00
0.8	97.00	97.00	96.00	96.00	96.00
0.9	97.00	96.00	96.00	96.00	96.00

Table A.4: Average packet delivery rate of FARINSECTOR for different values of π and α, in terms of percentages, for transmission radius $r = 15$m and $n = 150$.

Appendix B

Simulation Results for Radius 18m

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	83.32	3.82	92.17	2.37	97.20	1.72	99.01	0.89
COMPASS	84.63	3.56	93.54	2.41	97.88	1.49	99.26	0.84
BEST2COMPASS-U	90.13	3.01	96.11	1.93	98.74	1.12	99.60	0.62
BEST2GREEDY-U	88.70	3.16	95.09	2.02	98.29	1.34	99.38	0.76
FARINSECTOR	90.34	3.09	97.27	1.69	99.27	0.81	99.80	0.40
GREEDYINSECTOR	90.52	2.89	97.28	1.61	99.30	0.86	99.90	0.30
RANDOMINSECTOR	90.15	2.93	97.37	1.63	99.27	0.85	99.86	0.40
WEIGHTEDINSECTOR	90.56	3.28	97.31	1.64	99.31	0.79	99.87	0.37
WEIGHTEDSECTOR	91.94	2.91	97.64	1.47	99.36	0.81	99.88	0.36
AB:COMPASS-D-DET	84.15	3.62	92.83	2.23	97.61	1.52	99.14	0.90
AB:GREEDY-A-DET	83.78	3.98	92.83	2.33	97.47	1.59	99.03	0.94
AB:COMPASS-U	95.31	2.02	99.09	0.98	99.88	0.33	99.99	0.10
AB:GREEDY-U	94.70	2.38	98.90	1.07	99.79	0.43	99.99	0.10
AB:COMPASS-D	95.65	2.10	99.26	0.93	99.88	0.33	99.99	0.11
AB:GREEDY-D	94.99	2.02	98.71	1.17	99.75	0.48	99.98	0.14
AB:COMPASS-A	95.02	2.13	98.77	1.15	99.78	0.42	99.97	0.17
AB:GREEDY-A	93.37	2.50	98.20	1.33	99.45	0.73	99.90	0.30
INSECTOR:AB-U	91.57	2.76	97.70	1.47	99.38	0.78	99.90	0.30
INSECTOR:AB-A	91.59	2.84	97.56	1.53	99.42	0.81	99.87	0.34
INSECTOR:AB-D	91.80	2.65	97.64	1.51	99.43	0.74	99.88	0.33

Table B.1: Average packet delivery rate and standard deviation in *UDG*, in terms of percentages, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.02	0.06	1.02	0.06	1.01	0.06	1.01	0.05
COMPASS	1.08	0.13	1.09	0.14	1.10	0.15	1.11	0.16
BEST2COMPASS-U	1.53	0.97	1.34	0.67	1.24	0.42	1.19	0.28
BEST2GREEDY-U	1.48	0.94	1.28	0.60	1.17	0.44	1.11	0.25
FARINSECTOR	1.78	1.76	1.58	1.65	1.37	1.27	1.21	0.81
GREEDYINSECTOR	1.92	1.97	1.68	1.86	1.42	1.40	1.23	0.99
RANDOMINSECTOR	2.08	2.12	1.83	1.98	1.58	1.62	1.36	1.04
WEIGHTEDINSECTOR	2.03	1.99	1.83	1.96	1.57	1.59	1.37	1.08
WEIGHTEDSECTOR	1.72	1.54	1.56	1.43	1.39	1.12	1.28	0.84
AB:COMPASS-D-DET	1.05	0.10	1.05	0.11	1.05	0.11	1.05	0.11
AB:GREEDY-A-DET	1.04	0.09	1.04	0.10	1.04	0.10	1.04	0.10
AB:COMPASS-U	2.35	2.15	1.92	1.78	1.58	1.27	1.40	0.87
AB:GREEDY-U	2.27	2.15	1.85	1.68	1.52	1.28	1.33	0.90
AB:COMPASS-D	1.91	1.62	1.64	1.39	1.40	0.83	1.29	0.66
AB:GREEDY-D	1.88	1.66	1.60	1.41	1.34	0.87	1.23	0.70
AB:COMPASS-A	1.55	1.24	1.36	0.90	1.24	0.59	1.20	0.50
AB:GREEDY-A	1.49	1.23	1.36	1.22	1.21	0.82	1.14	0.56
INSECTOR:AB-U	1.70	1.51	1.55	1.35	1.41	1.16	1.29	0.85
INSECTOR:AB-A	1.70	1.51	1.55	1.38	1.39	1.17	1.28	0.80
INSECTOR:AB-D	1.74	1.56	1.59	1.50	1.42	1.07	1.31	0.70

Table B.2: Average stretch factor and standard deviation in UDG, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	82.26	3.66	91.15	2.55	96.60	1.83	98.56	1.08
COMPASS	83.30	3.75	92.22	2.36	97.12	1.60	98.76	1.07
BEST2COMPASS-U	90.52	2.89	96.39	1.77	98.66	1.21	99.60	0.57
BEST2GREEDY-U	88.99	3.21	95.54	2.09	98.46	1.23	99.52	0.67
FARINSECTOR	90.12	3.12	96.68	1.50	98.99	0.95	99.81	0.42
GREEDYINSECTOR	90.13	3.06	96.73	1.66	99.09	0.93	99.80	0.45
RANDOMINSECTOR	89.99	2.92	96.76	1.72	99.12	0.99	99.83	0.40
WEIGHTEDINSECTOR	89.97	3.08	96.79	1.66	99.13	0.98	99.87	0.37
WEIGHTEDSECTOR	91.51	2.75	97.11	1.48	99.16	0.85	99.85	0.39
AB:COMPASS-D-DET	82.69	3.50	91.59	2.45	96.87	1.72	98.70	1.07
AB:GREEDY-A-DET	82.88	3.93	91.78	2.37	96.82	1.83	98.65	1.17
AB:COMPASS-U	94.01	2.18	98.53	1.33	99.75	0.46	99.95	0.22
AB:GREEDY-U	93.40	2.31	98.50	1.26	99.62	0.62	99.95	0.22
AB:COMPASS-D	94.75	2.06	98.92	1.00	99.78	0.44	99.97	0.17
AB:GREEDY-D	94.28	2.47	98.65	1.13	99.61	0.62	99.94	0.24
AB:COMPASS-A	94.00	2.23	98.35	1.18	99.57	0.71	99.93	0.26
AB:GREEDY-A	93.00	2.31	97.99	1.39	99.38	0.75	99.89	0.31
INSECTOR:AB-U	90.99	3.11	97.18	1.62	99.13	0.86	99.88	0.33
INSECTOR:AB-A	91.14	2.70	97.18	1.53	99.10	0.93	99.82	0.41
INSECTOR:AB-D	90.99	2.81	97.23	1.61	99.12	0.92	99.87	0.37

Table B.3: Average packet delivery rate and standard deviation in *Yao* graph, in terms of percentages, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	77.54	4.28	86.50	3.13	92.65	2.52	95.29	2.30
COMPASS	77.28	4.17	86.10	3.32	91.86	2.72	95.14	2.37
BEST2COMPASS-U	88.47	3.35	95.88	2.01	98.56	1.24	99.24	0.85
BEST2GREEDY-U	88.24	2.87	95.69	1.89	98.45	1.27	99.35	0.69
FARINSECTOR	86.52	3.48	94.30	2.10	97.71	1.45	98.94	1.11
GREEDYINSECTOR	86.62	3.67	94.48	2.24	97.81	1.37	98.94	1.00
RANDOMINSECTOR	86.12	3.39	94.37	2.27	97.74	1.40	98.93	0.95
WEIGHTEDINSECTOR	86.91	3.58	94.37	2.29	97.70	1.38	98.78	0.96
WEIGHTEDSECTOR	88.52	3.48	95.10	2.06	97.86	1.51	98.90	1.06
AB:COMPASS-D-DET	77.61	4.32	86.50	3.12	92.68	2.55	95.33	2.23
AB:GREEDY-A-DET	77.26	4.13	86.05	3.35	91.89	2.65	95.12	2.40
AB:COMPASS-U	88.99	3.07	96.20	1.87	99.06	0.92	99.68	0.57
AB:GREEDY-U	89.09	2.85	96.50	1.89	98.93	0.95	99.70	0.59
AB:COMPASS-D	91.13	2.99	97.27	1.52	99.12	0.91	99.76	0.47
AB:GREEDY-D	91.14	2.90	97.08	1.59	99.12	1.00	99.77	0.51
AB:COMPASS-A	91.08	3.04	96.75	1.65	98.94	1.01	99.62	0.56
AB:GREEDY-A	90.84	2.73	96.72	1.83	98.83	1.06	99.65	0.56
INSECTOR:AB-U	88.04	3.30	95.20	1.98	97.97	1.36	99.02	1.00
INSECTOR:AB-A	88.23	2.95	94.97	2.00	97.87	1.33	99.04	0.86
INSECTOR:AB-D	88.06	3.13	95.14	2.04	97.90	1.34	99.00	1.02

Table B.4: Average packet delivery rate and standard deviation in *Gabriel* graph, in terms of percentages, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	83.23	3.68	92.68	2.92	96.84	1.80	98.95	0.97
COMPASS	83.93	3.77	93.41	2.80	97.17	1.51	99.19	0.92
BEST2COMPASS-U	90.74	2.90	96.65	1.90	98.84	1.08	99.71	0.52
BEST2GREEDY-U	89.82	3.13	96.07	2.01	98.58	1.16	99.57	0.66
FARINSECTOR	90.96	2.96	97.17	1.82	99.30	0.83	99.83	0.40
GREEDYINSECTOR	90.88	3.09	97.28	1.80	99.35	0.77	99.91	0.29
RANDOMINSECTOR	90.94	2.92	97.37	1.85	99.24	0.81	99.87	0.37
WEIGHTEDINSECTOR	91.15	3.02	97.26	2.07	99.34	0.95	99.88	0.33
WEIGHTEDSECTOR	92.11	2.61	97.58	1.65	99.36	0.81	99.84	0.39
AB:COMPASS-D-DET	83.60	3.75	92.95	2.71	97.01	1.72	99.10	0.98
AB:GREEDY-A-DET	83.74	3.78	93.24	3.00	97.11	1.50	99.07	0.92
AB:COMPASS-U	94.48	2.01	98.95	0.93	99.86	0.40	99.99	0.10
AB:GREEDY-U	93.71	2.50	98.72	1.18	99.70	0.56	99.97	0.17
AB:COMPASS-D	95.64	2.15	99.17	0.93	99.81	0.44	99.97	0.17
AB:GREEDY-D	94.49	2.24	98.89	1.15	99.73	0.49	99.95	0.22
AB:COMPASS-A	94.55	2.36	98.65	1.29	99.66	0.61	99.99	0.10
AB:GREEDY-A	93.54	2.49	98.24	1.44	99.45	0.73	99.92	0.27
INSECTOR:AB-U	92.04	2.62	97.75	1.55	99.36	0.76	99.89	0.31
INSECTOR:AB-A	91.74	2.38	97.69	1.53	99.34	0.82	99.88	0.33
INSECTOR:AB-D	91.82	2.60	97.48	1.67	99.33	0.80	99.88	0.33

Table B.5: Average packet delivery rate and standard deviation in *PLDel* graph, in terms of percentages, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.11	0.17	1.15	0.19	1.19	0.22	1.23	0.23
COMPASS	1.16	0.20	1.21	0.23	1.27	0.26	1.32	0.29
BEST2COMPASS-U	1.87	1.44	1.68	1.05	1.57	0.82	1.54	0.63
BEST2GREEDY-U	1.82	1.28	1.67	1.13	1.54	0.75	1.51	0.63
FARINSECTOR	2.07	1.95	1.98	2.10	1.79	1.54	1.69	1.33
GREEDYINSECTOR	2.22	2.25	2.05	2.01	1.86	1.73	1.71	1.29
RANDOMINSECTOR	2.24	2.11	2.14	2.09	1.95	1.80	1.82	1.46
WEIGHTEDINSECTOR	2.27	2.16	2.14	2.08	1.95	1.79	1.82	1.51
WEIGHTEDSECTOR	1.84	1.57	1.74	1.46	1.65	1.35	1.60	1.00
AB:COMPASS-D-DET	1.13	0.18	1.17	0.21	1.22	0.23	1.25	0.25
AB:GREEDY-A-DET	1.14	0.19	1.19	0.22	1.24	0.24	1.28	0.26
AB:COMPASS-U	2.62	2.36	2.30	2.17	2.00	1.74	1.81	1.29
AB:GREEDY-U	2.54	2.21	2.28	2.19	1.95	1.61	1.78	1.24
AB:COMPASS-D	2.16	1.80	1.95	1.61	1.74	1.17	1.65	0.95
AB:GREEDY-D	2.10	1.78	1.94	1.71	1.74	1.30	1.63	0.99
AB:COMPASS-A	1.68	1.34	1.61	1.28	1.49	0.96	1.46	0.72
AB:GREEDY-A	1.66	1.32	1.57	1.28	1.48	1.01	1.44	0.88
INSECTOR:AB-U	1.84	1.59	1.76	1.52	1.66	1.24	1.61	1.04
INSECTOR:AB-A	1.85	1.58	1.76	1.52	1.64	1.20	1.60	1.13
INSECTOR:AB-D	1.83	1.52	1.79	1.53	1.68	1.24	1.63	1.00

Table B.6: Average stretch factor and standard deviation in *Yao* graph, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.33	0.32	1.45	0.37	1.55	0.42	1.65	0.45
COMPASS	1.34	0.33	1.46	0.38	1.57	0.43	1.68	0.47
BEST2COMPASS-U	3.39	2.82	3.41	2.93	3.23	2.63	3.15	2.34
BEST2GREEDY-U	3.42	2.92	3.42	2.96	3.30	2.73	3.24	2.53
FARINSECTOR	2.86	2.65	3.07	2.78	3.12	2.89	3.20	2.83
GREEDYINSECTOR	2.96	2.71	3.14	2.92	3.21	3.03	3.26	2.96
RANDOMINSECTOR	2.95	2.64	3.15	2.93	3.20	2.84	3.23	3.03
WEIGHTEDINSECTOR	2.97	2.64	3.15	2.94	3.22	2.98	3.18	2.89
WEIGHTEDSECTOR	2.24	1.84	2.37	2.14	2.33	1.78	2.38	1.78
AB:COMPASS-D-DET	1.33	0.32	1.44	0.37	1.55	0.42	1.65	0.46
AB:GREEDY-A-DET	1.34	0.33	1.46	0.38	1.57	0.43	1.68	0.47
AB:COMPASS-U	3.55	3.05	3.73	3.30	3.68	3.29	3.61	3.07
AB:GREEDY-U	3.64	3.16	3.82	3.47	3.71	3.28	3.61	3.12
AB:COMPASS-D	2.97	2.48	3.15	2.70	3.11	2.60	3.09	2.44
AB:GREEDY-D	3.01	2.49	3.16	2.74	3.13	2.68	3.09	2.39
AB:COMPASS-A	2.19	1.83	2.24	1.99	2.20	1.77	2.21	1.57
AB:GREEDY-A	2.18	1.77	2.23	1.88	2.21	1.79	2.20	1.48
INSECTOR:AB-U	2.20	1.77	2.36	2.01	2.35	1.84	2.34	1.56
INSECTOR:AB-A	2.22	1.78	2.35	2.06	2.36	1.87	2.38	1.77
INSECTOR:AB-D	2.21	1.81	2.32	1.95	2.36	1.94	2.41	1.78

Table B.7: Average stretch factor and standard deviation in *Gabriel* graph, for transmission radius $r = 18$m.

	$n = 75$		$n = 100$		$n = 125$		$n = 150$	
Algorithms	Aver.	σ	Aver.	σ	Aver.	σ	Aver.	σ
GREEDY	1.14	0.18	1.21	0.22	1.28	0.25	1.35	0.28
COMPASS	1.17	0.21	1.26	0.25	1.33	0.28	1.41	0.32
BEST2COMPASS-U	1.92	1.37	1.76	1.02	1.70	0.76	1.70	0.59
BEST2GREEDY-U	1.90	1.34	1.73	0.97	1.68	0.72	1.68	0.61
FARINSECTOR	2.16	2.05	2.02	1.80	1.94	1.75	1.87	1.34
GREEDYINSECTOR	2.23	2.05	2.10	2.04	1.97	1.69	1.87	1.33
RANDOMINSECTOR	2.27	2.10	2.16	2.06	2.02	1.71	1.93	1.37
WEIGHTEDINSECTOR	2.29	2.18	2.12	1.90	2.03	1.79	1.92	1.24
WEIGHTEDSECTOR	1.83	1.52	1.76	1.36	1.70	1.17	1.68	1.02
AB:COMPASS-D-DET	1.15	0.19	1.22	0.22	1.29	0.26	1.36	0.29
AB:GREEDY-A-DET	1.17	0.20	1.25	0.24	1.32	0.28	1.40	0.31
AB:COMPASS-U	2.55	2.20	2.30	2.05	2.02	1.51	1.89	1.07
AB:GREEDY-U	2.59	2.33	2.29	2.01	2.04	1.58	1.89	1.09
AB:COMPASS-D	2.15	1.81	1.97	1.52	1.81	1.12	1.75	0.80
AB:GREEDY-D	2.13	1.75	1.96	1.53	1.81	1.25	1.75	0.87
AB:COMPASS-A	1.70	1.32	1.59	1.11	1.54	0.90	1.54	0.55
AB:GREEDY-A	1.67	1.28	1.58	1.15	1.54	0.91	1.55	0.78
INSECTOR:AB-U	1.86	1.55	1.78	1.44	1.71	1.22	1.68	0.93
INSECTOR:AB-A	1.85	1.57	1.74	1.37	1.69	1.16	1.69	0.99
INSECTOR:AB-D	1.86	1.59	1.77	1.40	1.70	1.11	1.70	0.94

Table B.8: Average stretch factor and standard deviation in *PLDel* graph, for transmission radius $r = 18$m.

Appendix C

Dynamic Maintenance of Fixed α in FARINSECTOR

In FARINSECTOR we always ensure that α is fixed, that is, the ratio of the area of the $Periphery(c,d,\theta,\alpha)$ to the area of $Sector(c,d,\theta)$ is always fixed, no matter how apart the current node c and the destination d are. In this section we present the calculation that we use for this purpose.

Let the current node c, the destination d, the transmission radius r, and the angle θ are given. We want to maintain a fixed value of α. First, we consider some pre calculations to find out the depending parameters of α. After that we show that three different cases might occur to keep the α fixed.

Let the coordinates of c and d be (x_1,y_1) and (x_2,y_2), respectively. Also, A and B be the intersection points between the $disk(c,r)$ and $disk(d,R)$, respectively. We consider Figure C.1 for the initial calculation.

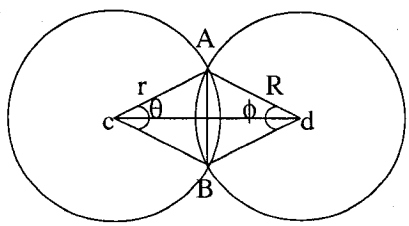

Figure C.1: The ratio of the area of $Periphery(c,d,\theta,\alpha)$ to the area of $Sector(c,d,\theta)$ is exactly α.

The distance L between c and d and the perpendicular bisector d of L are $\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ and $2r\sin\theta/2$, respectively. The hight of the $\triangle AcB$ is $h = r\cos\theta/2$. Then, R can be expressed as follows.

$$R = \sqrt{r^2+L^2-2Lr\cos\theta/2} \tag{1}$$

Also, the area of the $Sector(c,d,\theta)$ and the $\triangle AcB$ are $0.5r^2\theta$ and $r^2\sin(\theta/2)\cos(\theta/2)$, respectively. Hence, the area of the half-lune (inside $\triangle AdB$) ,denoted by A_{hl}, is as follows.

$$A_{hl} = 0.5r^2\theta - r^2\sin(\theta/2)\cos(\theta/2). \tag{2}$$

Similarly, the area of the $Sector(d,c,\phi)$ and the $\triangle AdB$ are $0.5R^2\phi$ and $r\sin(\theta/2)(L - r\cos\theta/2)$, respectively; where ϕ can be expressed as $2\arcsin\frac{d}{2R}$. The area of the half-lune (inside $\triangle AcB$), denoted by B_{hl}, is as follows.

$$B_{hl} = 0.5R^2\phi - r\sin(\theta/2)(L - r\cos\theta/2). \tag{3}$$

Thus the total area of the lune, denoted by $Area_{lune}$, is $A_{hl} + B_{hl}$

C.1 Three Cases of α

We calculate the ratio of $Area_{lune}$ to the sector's area with the given values, and check in which of the following cases it fits.

Case 1: The ratio of the area of *lune* to $Sector(c,d,\theta)$ is equal to α, which can be expressed as $\frac{Area_{lune}}{0.5r^2\theta} = \alpha$.

We are in this case only when the calculated ratio is just same as α, and we can directly use all the above mentioned parameters without any adjustment.

Case 2: The ratio of the area of *lune* to $Sector(c,d,\theta)$ is greater than α, which can be expressed as $\frac{Area_{lune}}{0.5r^2\theta} > \alpha$.

In this case the intersected area is greater than α. To make the ratio fixed to α, we need to redefine R, ϕ, and θ. The *Periphery* and *Core* are defined later with the new value of R, called R'. We will again consider Figure C.2 for further calculation.

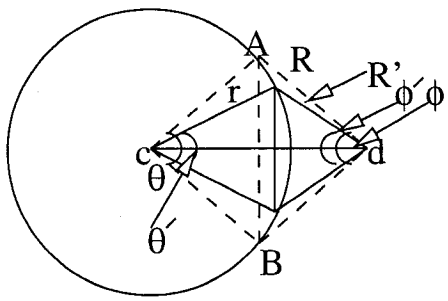

Figure C.2: The ratio of the area of $Periphery(c,d,\theta,\alpha)$ to the area of $Sector(c,d,\theta)$ is greater than α.

From the above figure the area of the lune can be written as follows.

$$Area_{lune} = (0.5r^2\theta' - 0.5h'd') + (0.5R'^2\phi' - 0.5(L-h')d')$$

$$= 0.5r^2\theta' + 0.5R'^2\phi' - Lr\sin\theta'/2 \tag{4}$$

The ratio of $Area_{lune}$ to area of the sector should be equal to α, which gives us the following equation.

$$0.5r^2\theta' + 0.5R'^2\phi' - Lr\sin\theta'/2 = \alpha(0.5r^2\theta) \tag{5}$$

Again, $R' = L^2 + r^2 - 2Lr\cos\theta'/2$, which gives us new form equation 5 with a constant $C_0 = \alpha r^2\theta$.

$$r^2\theta' + (L^2 + r^2 - 2Lr\cos\theta'/2)\phi' - 2Lr\sin\theta'/2 - C_0 = 0 \tag{6}$$

Also, $\tan\phi'/2 = \frac{d'/2}{L-h'}$, where d' and h' is $2r\sin\theta'/2$ and $r\cos\theta'/2$, respectively. These values give us equation of ϕ' as follows.

$$\phi' = 2\arctan\frac{r\sin\theta'/2}{L - r\cos\theta'/2} \tag{7}$$

Now, by substituting the value of ϕ to equation 6, we will get the equation of θ'.

$$\theta' = \frac{C_0 - (L^2 + r^2 - 2Lr\cos\theta'/2)(2\arctan\frac{r\sin\theta'/2}{L-r\cos\theta'/2}) + 2Lr\sin\theta'/2}{r^2} \tag{8}$$

Also, we can rearrange it in the following form, and solve it using one-point *iterative*

method.

$$\theta' = f(\theta') \tag{9}$$

C.2 Solution of One Variable Algebraic Equation

We will briefly discuss numerical methods of solving equations of the form $f(x) = 0$, where both x and $f(x)$ are real. Real values of x for which $f(x) = 0$ holds are called *roots* of the equation. Lets write $f(x) = 0$ in the equation form for further reference.

$$f(x) = 0 \tag{10}$$

We need to evaluate $f(x)$ at enough points so that we can sketch the curve $y = f(x)$ with sufficient accuracy to locate all the roots of equation 10. After locating the roots approximately, we need to refining them. The refining methods should have the following properties. Beginning with an interval I_0 which contains at least one root of equation 10, we construct a sequence of intervals I_n such that $I_{n+1} \subset I_n$ and each I_n contains at least one root of equation 10. This is called a bracketing method (a root is bracketed by the endpoints of each interval I_n). Obviously, we would like the length of the interval I_n to tend to zero as $n \to \infty$ so that in principle we may compute the root as accurately as we wish [PT96].

One-point Iterative Method

Most of the numerical methods generally use at least two values of x in order to predict a better approximation to the root, at any stage. If a method use only one value of x at any time, then it is called one-point method. To find a root α_r of $f(x) = 0$, we need to construct a sequence(x_r) which satisfies two criteria: 1. the sequence (x_r) converges to α_r, 2. x_{r+1} depends directly only on its predecessor x_r.

From 2, we need to find some function g so that the sequence x_r may be computed from

$$x_{r+1} = g(x_r), r = 0, 1, ... \tag{11}$$

given an initial value x_0. If we happened to choose $x_0 = \alpha_r$, we would also want 10 to

give $x_1 = \alpha_r$ and therefore g satisfies the property $\alpha_r = g(\alpha_r)$. Thus, to obtain a one-point method for finding a root α_r, of the equation 9, we rearrange the equation in a form $x = g(x)$, with $\alpha_r = g(\alpha_r)$, so that equation 11 yields a sequence (x_r) which converges to α_r [PT96].

It is clear from the above discussion that we can solve equation 9 using one-point method with an initial value of the root. In equation 9 this initial value of the root as 0.5. After getting the value θ', we can easily calculate ϕ' and R'. Hence, we can define *Periphery* and *Core* with these new values [PT96].

Case 3: The ratio of the area of *lune* to $Sector(c,d,\theta)$ is less than α, which can be expressed as $\frac{Area_{lune}}{0.5r^2\theta} < \alpha$.

In this case new R and ϕ will be more than the original values. We will consider Figure C.3 to show the adjustment. The area of the lune can be written as follows

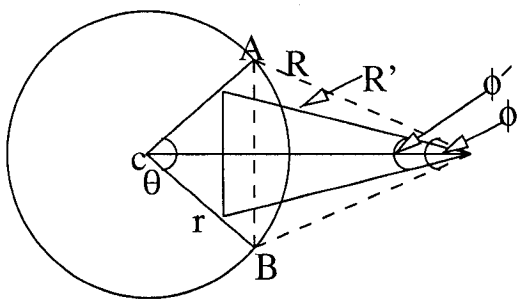

Figure C.3: The ratio of the area of $Periphery(c,d,\theta,\alpha)$ to the area of $Sector(c,d,\theta)$ is less than α.

$$Area_{lune} = 0.5r^2\theta + 0.5R'^2\phi' - 0.5Ld' \tag{12}$$

The ratio of $Area_{lune}$ to area of the sector should be equal to α, which gives us the following equation.

$$0.5r^2\theta + 0.5R'^2\phi' - 0.5Ld' = \alpha(0.5r^2\theta) \tag{13}$$

$$\Rightarrow 0.7r^2\theta + R'^2\phi' - Ld' = 0 \tag{14}$$

We get the following equation after using the value of R'^2 and d' as $L^2 + r'^2 - 2Lr'\cos\theta/2$

and $2r'\sin\theta/2$, respectively. The two constants are $C_1 = 0.7r^2\theta$ and $C_2 = 2L\sin\theta/2$.

$$0.7r^2\theta + (L^2 + r'^2 - 2Lr'\cos\theta/2)\phi' - (2L\sin\theta/2)r' = 0 \tag{15}$$

$$\Rightarrow C_1 + (L^2 + r'^2 - 2Lr'\cos\theta/2)\phi' - C_2 r' = 0 \tag{16}$$

Also, $\phi' = 2\arctan\frac{r'\sin\theta/2}{L - r'\cos\theta/2}$, and by substituting this value to equation 16, we will get the equation of r'.

$$C_1 + (L^2 + r'^2 - 2Lr'\cos\theta/2)2\arctan(r'\sin\theta/2/L - r'\cos\theta/2) - C_2 r' = 0 \tag{17}$$

Again we can express 17 as $r' = f(r')$, and can be solved it using one-point *iterative method.*

After getting all the new values we can define our *Periphery*, *Wing* (in case 3 we get wing outside the sector, this allows us to pick next node outside the sector.) and *Core* to get next candidate nodes.