Improving Routing Efficiency in Location-Aware Wireless Ad-hoc Networks

by

Paul J. Boone

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Masters of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

November 2003

The undersigned recommend to

The Faculty of Graduate Studies and Research

acceptance of the thesis,

Improving Routing Efficiency in Location-Aware Wireless Ad-hoc Networks

submitted by

Paul J. Boone

Dr. Doug Howe

(Director, School of Computer Science)

Dr. Evangelos Kranakis

(Thesis Supervisor)

Carleton University

November 2003

Abstract

This thesis considers the problem of location-aware routing in wireless ad-hoc networks. Location-aware algorithms operate in networks where nodes have a knowledge of their geographical location, and a geometric graph can be used to represent the underlying physical network. Our contributions focus on two areas in location-aware routing.

We introduce a new localized algorithm, the *Morelia Test*, to create a planar spanning subgraph of the graph representing an underlying wireless ad-hoc network. We show, through a simulation study, that the location-aware *face routing* algorithm performs better with the Morelia graph than with the *Gabriel graph* when the underlying network has regions which are sparsely populated.

We also develop enhancements to the face routing algorithm, including region avoidance, multiple transmission ranges, and route discovery.

Finally, we summarize our results and discuss some areas of future work that should be investigated.

Acknowledgements

I would like to thank my supervisor Dr. Evangelos Kranakis for his help and support of my work. I would also like to thank Simon MacDonald, Miguel Vargas Martin, Sylvain Pitre, David Whyte, my parents and sister, and finally my wife Jenny, for all their contributions to the completion of this thesis.

This research was supported in part by Natural Sciences and Engineering Research Council, NSERC.

Contents

List of Figures

Chapter 1

Introduction

We are about to enter a new information age. This will be brought about by the proliferation of next generation wireless technologies that will have the capability of providing *location-based* or *location-aware* services, which are services based on the information about a user's current location. Wireless device vendors promise to include multiple wireless interfaces in next generation devices. These include the standard cellular interface as well as interfaces such as a wireless local area network (WLAN) standard. Future wireless phones will have the ability to move freely between the traditional cellular networks and the WLAN networks that may be present in shopping malls or workplaces. A prime example of such an application lies in the enterprise environment where corporations could save millions of dollars by assigning to their employees a wireless phone that operates over the internal WLAN corporate network while they are in the office, but could seamlessly switch over to the cellular network when out of the office on business.

Effective November 24, 2003, the Federal Communications Commission in the United States will introduce new rules that wireless carriers must provide Wireless Local Number Portability [44], the ability for users to keep their current wireless cellular phone numbers when changing their service provider. Currently, there are millions of cell phones upgraded each year in the US and now, with the introduction of these new rules, this number is expected to increase dramatically in the next few years. This will pave the way for wireless carriers to introduce these *location-aware* services which have promise to be the suite of "killer-apps" for wireless networks. These *location-aware* services, from map and building directory services, to targeted advertising, may begin invading our everyday life in the very near future.

In this thesis we will study the problem of routing in wireless ad-hoc networks, focusing on *location-aware* routing algorithms. Location-aware algorithms take advantage of the fact that all nodes in the network have knowledge of their geographic location, using this information in decision making.

1.1 The Problem

Routing in a wireless communication system is a major problem that has been widely studied. Usually, devices in a network have little knowledge about their surrounding environment. Traditional algorithms try to give nodes a global knowledge by building routing tables or routes used in sending packets from a source node to a target node. However, if nodes know their location, the locations of their neighbours, and the location of the target, location-aware algorithms do not need to build such tables or routes and can make routing decisions locally.

In this thesis we will study routing in wireless ad-hoc networks, sometimes called Mobile Ad-hoc Networks (MANETs). We will see some traditional routing algorithms developed for ad-hoc networks, but will focus on *location-aware* routing algorithms giving various examples.

Our main focus will be the requirement of a *planar graph* for some location-aware routing protocols. We will present a new algorithm, called the *Morelia Test*, that can be used to produce a planar spanning subgraph of an underlying wireless network. A second area of study will be to provide enhancements to the *face routing* protocol.

1.2 General Model and Notation

A *graph* $G = (V, E)$ is defined as a set of n vertices V and a set of m edges E. An edge $e \in E$ is made up of a pair of vertices. A *planar* graph G is one where no two edges of G intersect at a point other than an endpoint at some vertex $v \in V$. The edges of G divide the graph into a set of polygons, called *faces*. The *neighbourhood* $N(v)$ of a vertex $v \in V$ is the set of vertices adjacent to vertex v. A network is *connected* if for every pair of vertices $u, v \in V$, a route can be found from u to v by traversing adjacent edges.

1.3 Motivation - Wireless Ad-hoc Networks

The main motivation behind investigating location-aware routing is in making wireless ad-hoc networks more efficient. Wireless ad-hoc networks or Mobile Ad-hoc Networks (MANETs) are comprised of a number of autonomous nodes which act as routers as well as hosts, collaborating to form a productive network.

The nodes form a network "on demand", without any fixed infrastructure, using wireless links and may move around in a random fashion. Figure 1.1 shows a wireless ad-hoc network and depicts the movement of mobile node 4. Nodes may connect to and disconnect from the network at any time. This results in an ever changing topology that is hard to predict. Due to this property of an ad-hoc network, the graph representing the network may be non-planar.

Two nodes can communicate in a bidirectional manner if and only if the distance between them is less than the minimum of their transmission ranges. If one node wishes to communicate with another node which is not in transmission range, it may

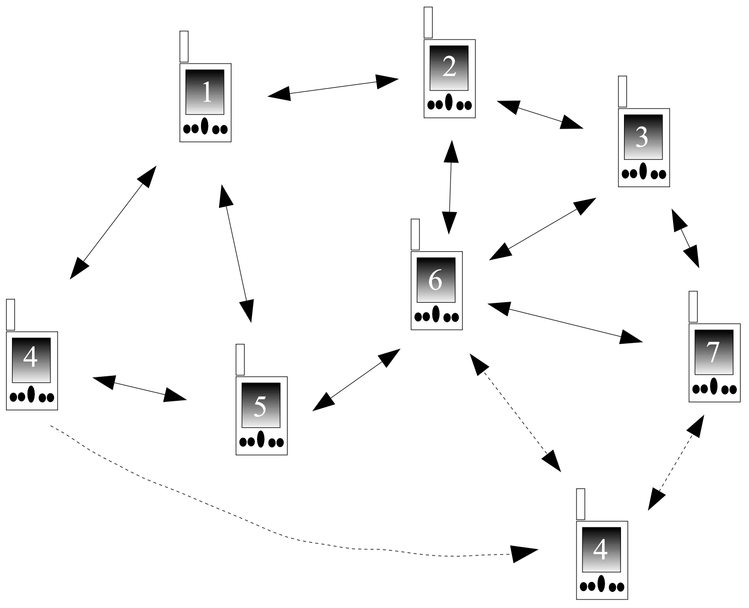

Figure 1.1: A Wireless Ad-hoc Network: Mobile Node 4 moves and this changes the network topology.

do so by forwarding messages through other nodes in the network in a multi-hop manner. Figure 1.2 shows that node 1 may communicate with node 3 by forwarding messages through node 2.

The *unit disk graph* (UDG) is a graph G where an edge (u, v) exists if the distance from u to v is ≤ 1. The UDG is what is typically used to model an ad-hoc network where all nodes in the network have the same transmission range. The nodes can be represented as V, the set of vertices, and there is an edge $e \in E$, between two nodes u and v if they are within communication range of each other. Most research in ad-hoc networks is based on nodes which have the same transmission range. There is, however, an interesting paper [1] where the authors deal with the fact that the transmission ranges are not identical.

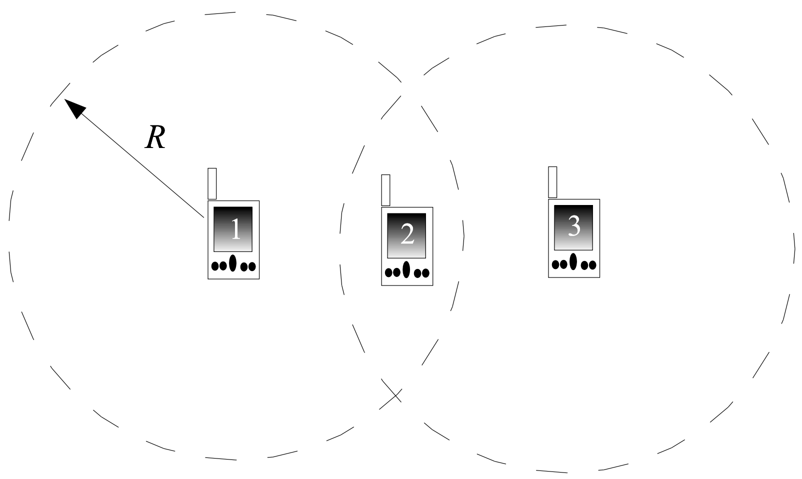

Figure 1.2: Multi-hop: Node 1 can exchange messages with Node 3 by relaying them via Node 2.

Scalability and Location Information

As mentioned earlier, traditional routing algorithms attempt to give nodes a global view of the network. This requires a large amount of storage and communication overhead. Nodes typically need to store an amount of information proportional to the size of the network. This will result in a threshold where at some point, more bandwidth will be consumed in maintaining this information than actually sending data.

The advantage of a location-aware algorithm is that no information is propagated around the network to maintain route information. Nodes in the network will only need to periodically broadcast their location information to inform their neighbours or their presence. This means that more bandwidth can be used in routing data instead of maintaining the global picture.

1.4 Overview

In Chapter 2, we will present an overview of the traditional (non location-aware) routing algorithms that have been developed for wireless ad-hoc networks. We will see that they are divided into three classes, *proactive, reactive and hybrid* protocols and will give some examples of these protocols.

In Chapter 3, we will define a model for location-aware routing algorithms. We will then give an overview of the development of location-aware routing algorithms. Beginning with simple Greedy Routing, advancing to more complex algorithms such as Greedy Perimeter State Routing, Face Routing and its variants, as well as Location-Aided Routing and Distance Routing Effect Algorithm for Mobility

In Chapter 4, we discuss existing *local* algorithms that can be used in preprocessing an underlying ad-hoc network to produce a *planar spanning subgraph* suitable for specific location-aware routing algorithms.

We begin with discussing the Relative Neighbourhood Graph and the Gabriel Graph. Then, we discuss a more recent work on computing the planar spanning subgraph which is based on computing a localized Delaunay triangulation of the network.

1.5 Contributions of the Thesis

In this thesis we make the following contributions in the area of routing in location-aware wireless ad-hoc networks.

1.5.1 The Morelia Test: Creating a better planar spanning subgraph

In Chapter 5 we will describe a new localized test developed to produce a planar spanning subgraph of an underlying ad-hoc network topology. This test, called the *Morelia Test*, is designed to improve the results obtained from routing on the *Gabriel Graph*. Routing will be improved when the underlying network has areas which are sparse. We show, that with only local information, including the nodes location, the location of its 1-hop neighbours, and some communication overhead, we can produce planar spanning subgraph suitable for various location-aware routing protocols. We will show our simulation results of the Morelia test in comparison with the Gabriel test.

1.5.2 Enhancements to the Face Routing Algorithm

In Chapter 6 we define three enhancements to the face routing algorithms [23, 6]. The first algorithm introduced is used to avoid a region in an ad-hoc network. The second algorithm is the application of a modified face routing for route discovery. The final two algorithms defined show a way to preserve energy consumption of the face routing algorithm.

Region Avoidance

The first section discusses *region avoidance* in wireless networks. There may be various reasons to avoid certain *geographical* regions of a network, one of the most important being security. In this section we present two approaches to the problem.

Face Routing for Route Discovery

In the next section, we change the *face routing* algorithm into a route discovery algorithm. The goal is to eliminate loops and reduce the multi-hop effect.

Energy Preservation with Multiple Radii

In the next section, we describe two simple algorithms to preserve energy consumption during face routing in an ad-hoc network if nodes have the ability to choose from a set of k different power levels (transmission ranges).

Chapter 2

Traditional Routing in Ad-hoc Networks

2.1 Introduction

In this chapter we study a suite of early algorithms (non location-based) developed to solve the routing problem in ad-hoc networks. These algorithms are divided into three classes, *proactive*, *reactive*, and *hybrid*.

A *proactive* routing protocol is one that attempts to keep all information about the network topology and routes from all sources to all destinations. These protocols resemble those used in a wired network. The advantage of a proactive routing protocol is that routes are readily available when one node wishes to send a message to another. One disadvantage of proactive routing is that the algorithm requires memory storage at each node of a size proportional to the size (number of nodes) of the network. A second disadvantage is that in order to maintain the routing information, a large number of network control messages are required to be sent around the network. In section 2.2 we will see the *Destination Sequenced Distance Vector* protocol.

A *reactive*, or on demand, routing protocol is one that tries to minimize the number of network control messages introduced on the network (less bandwidth used for maintenance means more bandwidth for routing data). These algorithms will only discover a route from a source to a target when the source wishes to communicate with the target. One advantage of these protocols is that nodes only need to store information on routes that they are currently using. A second advantage is, as mentioned earlier, these protocols try to use less messages for maintaining routing information. The main disadvantage of these algorithms is that if a node does not have a route to a target it wishes to communicate with, it must be discovered. This results in a delay before the first packet can be sent. However, once a route has been discovered, messages can be sent normally. In sections 2.3 and 2.4 we will see the *Ad-hoc On Demand Distance Vector* and *Dynamic Source Routing* protocols.

The final type of routing algorithms discussed here are *hybrid* protocols. These algorithms try to combine both proactive and reactive routing protocols to obtain the advantages of both. These protocols typically divide the ad-hoc network into regions and perform a proactive routing protocol in these regions, and a reactive routing protocol when routing between regions. In section 2.5 we will see the *Zone Routing Protocol.*

2.2 Destination Sequenced Distance Vector - DSDV

The Destination Sequenced Distance Vector (DSDV) protocol is a hop-by-hop distance vector protocol [28, 37] that enables multi-hop routing between nodes in an ad-ho network. Each node in the network must store a routing table containing all destinations it can route to. The information is the next-hop neighbour, the number of hops (hop-count) to reach the destination, as well as a sequence number. DSDV requires that all nodes periodically broadcast their routing information, or do so immediately (trigger) when a change to this information occurs. DSDV is a loop free protocol, and to achieve this, the protocol uses the sequence number mentioned earlier to mark routes. The sequence number is an indication of how old the route is. The higher the sequence number, the more current the information and these routes are more favorable. A route x would be considered more favorable that another route y if the sequence number of route x was higher than that of y or their sequence numbers are the same, but route x has a lower hop-count. When a node discovers that a route in its table has been broken, it will increase the sequence number and set the hop-count to ∞. The node will then advertise its routing information which may trigger updates to neighbouring nodes as well.

The DSDV protocol is the distance vector protocol with a few modifications to make it more suitable for use in ad-hoc networks. The addition of triggering updates when changes occur in the network topology, ensure that this information is distributed to all the nodes in the network. DSDV requires a lot of control messages to update route information across the network. This has led to defining two types of update messages.

1. **Full dump**: Message carries the entire routing table for a node.
2. **Incremental Dump**: Message carries only the changes in the routing table for a node.

One of the main drawbacks of DSDV, as well as most proactive routing protocols for ad-hoc networks, is due to the fact that information must be gathered for all routes in the network. This means it will take some time for the information to converge and produce a stable routing environment. Also, at each node, routes are kept for all destinations in the network. This can result in a lot of storage overhead within each node.

2.3 Ad-hoc On Demand Distance Vector - AODV

The Ad-hoc On Demand Distance Vector (AODV) protocol [28, 35] is one that provides multi-hop routing in ad-hoc networks. It is also based on the distance vector routing protocol. The main advantage of AODV is that is it a reactive protocol, unlike DSDV which is a proactive one. This means that AODV only discovers routes to destinations for which it is currently trying to use. This may allow for a dramatic reduction of storage space required at each node.

Main advantages of AODV are reduced storage of route information, loop free routing, and immediate notification of links which become disconnected. AODV, like DSDV, also uses sequence numbers to ensure that routes remain fresh.

The protocol uses several control messages to gather routing information and maintain links. When one node wishes to discover a route to another node it will broadcast a Route Request (RREQ) message which is heard by its neighbours. This RREQ will propagate throughout the network until reaching the destination node, or a node on the way which has the route information the source node is looking for. When the RREQ is captured, a Route Reply (RREP) message is sent back to the source node using unicast. To maintain information, nodes will periodically broadcast a beacon so that their immediate neighbours are aware of their continued participation in the network. The neighbours that are using the node (from which they received a beacon) for routing can keep the routes as valid. If a neighbour fails to receive a beacon from one of its neighbours after a time, it will mark the link to that neighbour as disconnected, and send a message indicating the link has failed to the affected nodes which use that link for routing.

2.3.1 Route Discovery in AODV

When a node needs to discover a route for a destination it does not have, it will broadcast a RREQ message. This may occur as the node has no route to that destination, or the route may have expired or it may have been notified that it has failed. The node will wait until it receives a RREP message. If no RREP received with in a fixed timeout period, the node will retry a set number of times and then assume there is no route to this destination.

When a node receives a RREQ message for destination x, one of two things may occur.

1. The node does not have the route information: In this case the node will re-broadcast the RREQ and will also archive (temporarily) the reverse path back to the source node for possible future use.

2. The node is x or has route information for x: In this case, a RREP can be sent directly back to the source via unicast. As the RREP is forwarded back to the source node, routing information is updated along the route. When the source receives the RREP, a route to the destination x will have been created and can be used.

2.3.2 Route Maintenance in AODV

If a node realizes that a neighbour is no longer present, it must remove the routing information for that neighbour. It will then send a link failure message to the neighbours in the active neighbour list for that destination. All nodes that receive this message will repeat the procedure and the information in the network will be corrected. Any node sending information to a destination over this route will be required to rediscover a route.

2.3.3 Features

The biggest advantage of AODV over wire-based routing protocols such as distance vector is that AODV, due to its reactive nature, greatly reduces the number of network control messages introduced. AODV also more closely resembles wire-based protocols, as opposed to DSR (in the next section) and this may make integrating an ad-hoc network with a wired network easier.

2.4 Dynamic Source Routing - DSR

Dynamic Source Routing (DSR) [17, 28] is another reactive routing protocol for ad-hoc networks. It differs from the distance vector protocols in the sense than the source node knows the hop-by-hop route that must be taken by the packet from source to destination. This route is stored in the packet header and intermediate nodes along the path simply forward the packet to their neighbour which is next on the path.

The DSR protocol is made up of two stages. The first is *route discovery* and the second is *route maintenance. Route Discovery* is used when a source node s wishes to send a packet to a destination node d but does not have such a route stored in memory. It is a mechanism to find the source route from source to the destination. *Route Maintenance* is a mechanism where node s can detect, while sending a packet over the source route to d, that the network nodes have moved in such a way that prevent it from using this route. Some link along the route has been broken, and s can use an alternate source route to d, or begin the route discovery mechanism again. Route maintenance is only used while sending packets from s to d.

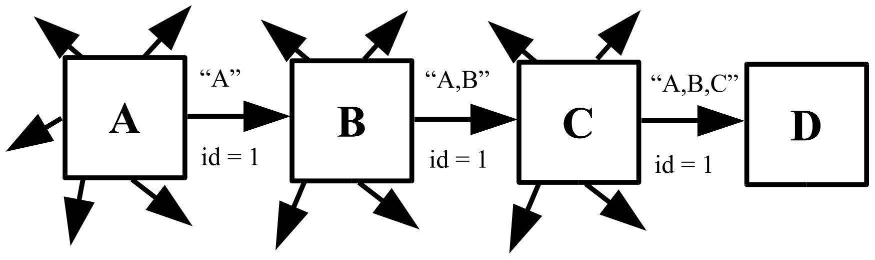

Figure 2.1: An example of DSR route discovery. A is initiator, D is destination.

2.4.1 Route Discovery

To discover a route to the destination, the sender *s* will initiate a route discovery by sending a RREQ message to its neighbours. The RREQ message identifies the sender and the destination node and also contains a unique *request identifier* chosen by the sender. The RREQ message also contains a list of addressed of all intermediate nodes the packet has been forwarded through. Figure 2.1 shows and example of the route discovery procedure with node A as the initiator and node D as the destination. When a node receives a RREQ packet, one of two things will occur.

1. If the node is the intended destination, it will rend a RREP message back to the initiating sender. In the reply, a copy of the source route is included so the sender can cache this route for future use.

2. If the node is not the intended destination, but it has seen a RREQ from the same source with this request identifier, or it sees its own address in the route information it will simply ignore the message and discard it. If the request is new, then the node will add its address to the end of the route and forwards the request with the same request identifier to its neighbours as a local broadcast.

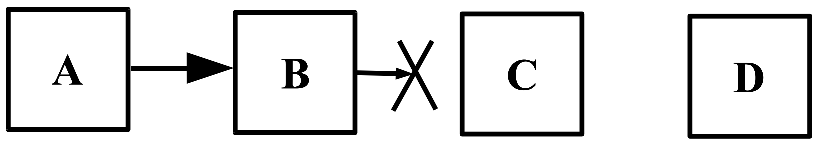

Figure 2.2: An example of DSR route maintenance. A wants to send a packet to D, but the link from Node B to Node C is broken.

2.4.2 Route Maintenance

In DSR, route maintenance is handled while nodes are forwarding packets over the source route. Each node forwarding a packet is responsible to make sure the next-hop node has received this packet. This can usually be done with little overhead to the DSR protocol via the IEEE 802.11 link-level acknowledgement frame for example [16]. A second method for a node to know that a packet has been received by the next-hop node is by a *passive acknowledgement* [18] where the forwarding node overhears the next-hop node forwarding the packet to its next-hop node. If the forwarding node cannot use one of these two methods, there is a third alternative. The forwarding node can set a bit in the packet header requesting a DSR acknowledgement to be sent back from the next-hop node.

If a packet is not acknowledged, it will be resent for a set number of retries. If it still has not been acknowledged, the forwarding node will assume there is a problem and generate a Route Error (RERR) message to be sent back to the source sender of the packet. This RERR will let the sender know that the route has failed, and which specific hop along the route caused the trouble. In figure 2.2 we see that B cannot send a packet to C. In this case B will send a RERR message back to A informing A

that link BC is broken. Node A will then remove this hop link from its route cache. If A wants to send additional packets to destination D, it will have to either find an existing alternate route in its cache or reinitiate a route discovery procedure for destination D.

2.4.3 Features

The main advantage of source routing is that intermediate nodes in the network used to forward packets are not required to maintain up to date routing information. In DSR, there is also no need for periodic beacons to update neighbourhood information. This will reduce bandwidth and power overhead, especially when there is little or no node mobility.

A node can also discover route information by listening in on packets that it receives. If, for instance a node A routing packets to node C via node B will discover both routes to both B and C.

The fact that packets must include the source route from source to desitnation node means that each and every packet introduces some bandwidth overhead to store this route information. This overhead is directly proportional to the number of hops packets must traverse from source to destination.

Finally, in DSR, the nodes operate in promiscuous mode, listening in to all packets it receives scanning for information. This is a security risk and some mechanism must be provided to secure the protocol (i.e. some form of encryption).

2.5 Zone Routing Protocol - ZRP

The Zone Routing Protocol (ZRP) [14, 28] is in a class of *hybrid* routing protocols. It combines both a reactive and proactive routing protocol. The network is subdivided into regions, called *zones* and has a protocol for routing inside a zone and one for routing between multiple zones.

The two protocols are the Intrazone Routing Protocol (IARP) and the Interzone Routing Protocol (IERP). IARP is used to route packets within a zone, and may use a *proactive* routing protocol such as DSDV. Different zones may run different protocols. Also, when mobility results in a topology change, this information is only propagated within the affected zone and not throughout the entire network.

The IERP is a *reactive* routing protocol and used for discovering routes across the network between multiple routing zones. If the destination is not in the same routing zone, the protocol will broadcast a RREQ message to all border nodes in the current zone (bordercasts). The border nodes will forward the RREQ message if the destination node is not in their routing zone. This repeats until the destination node has been found and a RREP is returned to the sender. The IERP includes the Bordercast Resolution Protocol (BRP) which handles management of nodes on the border of the zone. This protocol allows for the sending of messages from one node to all of the nodes at the edge of the current zone.

2.5.1 The Routing Zone

The routing zone of a node A is defined by the set of all nodes which can be reached within a specific minimum number of hops from A. In figure 2.3 we see that nodes A, B, C, E, H, I, J are all within 2 hops from node D. Also, suppose node A wishes to send a message to node G. Since G is not in the routing zone of A, a route request message is sent to the border nodes of *Zone A* (C and D). These nodes will check to see if G is in their routing zone. G is not found and thus C forwards the request to the next set of border nodes A, D, F and H. D forwards the request to A, C, E, I and J. Now, the destination node G is found in the zones of both E and F. A reply is sent back to node A.

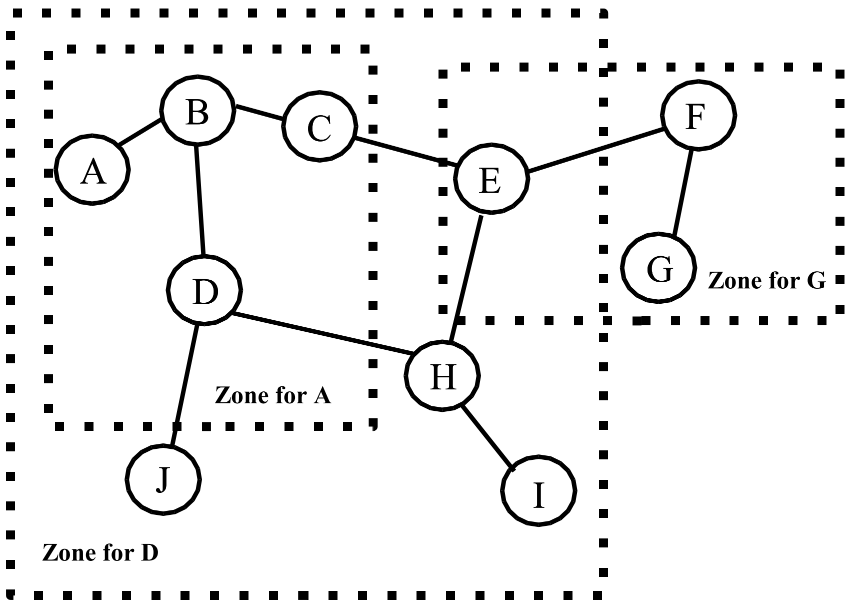

Figure 2.3: ZRP example - Shows zones for nodes A, D, and G.

2.5.2 Features

This protocol is a hybrid one taking advantage of both proactive and reactive aspects. Routes can be discovered very quickly within a zone, and destinations outside the routing zone are found by asking a subset of nodes from within the zone.

Another advantage of the Zone Routing Protocol is that the protocol can be adjusted to work on different topologies. The zone diameter (number of hops) can be changed by an administrator to optimize operation in particular network scenarios.

2.6 Overview

In this section, we have seen an overview of traditional routing algorithms for ad-hoc networks. The main disadvantage of these algorithms is that they require large overhead in maintaining routing tables or discovering routes. These protocols will "flood" the network during route discovery, or routing table updates.

Chapter 3

Location-Aware Routing

3.1 Introduction

In this chapter we describe a class of routing algorithms known as *location-aware* or *geometric* routing. This class of algorithms provide some additional advantages over traditional topology based algorithms. We will see these advantages in the description of the general model for geometric routing protocols. These algorithms make use of information about the physical or geographical position of the nodes in the network. In the following sections we will present a general model and a brief survey of location-aware routing algorithms. With the exception of the final two algorithms presented, these algorithms do not "flood" the network during their operation, but select the next "best" node to forward packets. LAR and DREAM used *partial flooding* to forward packets within a region of the network defined by the location of the sending node and the expected range of location of the target. The main disadvantage of location-aware algorithms over traditional algorithms is the trade-off of memory versus optimal routes. Since traditional algorithms propagate information about the network to the nodes in the network, more optimal routes can be found.

3.2 The Model

The following describes the general characteristics of a model for a location-aware routing algorithm.

1. Each node in the network has the means to determine their physical position in their environment. This can be obtained through a GPS receiver or other such mechanism.

2. All nodes must make use of a *location service* to find the position of a device with which they wish to communicate. *Location Services* are not discussed here.

3. With the location information of the destination, the routing decisions at each hop in the route can be made based on the location of the current node and the location of the destination

4. There is no need for devices to store routing tables. Devices only maintain the information about their one-hop (sometimes two-hop) neighbours.

5. There is no need for additional messages sent in the network to maintain routing tables. Devices in the network will simply periodically broadcast a *beacon* containing information such as node identifier and geographic coordinates.

6. An additional advantage of using routing based on position, is that it allows for the ability to send messages to devices in a particular area. An example of this would be a targeted advertising application where information is sent to users as they enter a certain area such as a shopping mall. This is called *geocasting* [33] and will not be discussed further.

However, there are many variations on this model. For example, some algorithms may choose to store some information on routes that have discovered such as LAR (which we will see shortly) only using location information for route discovery and not for actual routing. Some algorithms may not explicitly use a location service, but may propagate location information about nodes within data packets.

3.3 Traditional Greedy Routing

In this section we will deal with a suite of *greedy* algorithms [13, 31]. But first, we will introduce the definition of *progress* [13]. Progress is the main idea behind several routing algorithms. If we have a node s wishing to forward a packet, the progress of a neighbour node n is the projection onto the line from s to the destination. If node n lay in the forward direction, then the node has *positive* or *forward progress*, otherwise it is said to have *negative* or *backwards progress*.

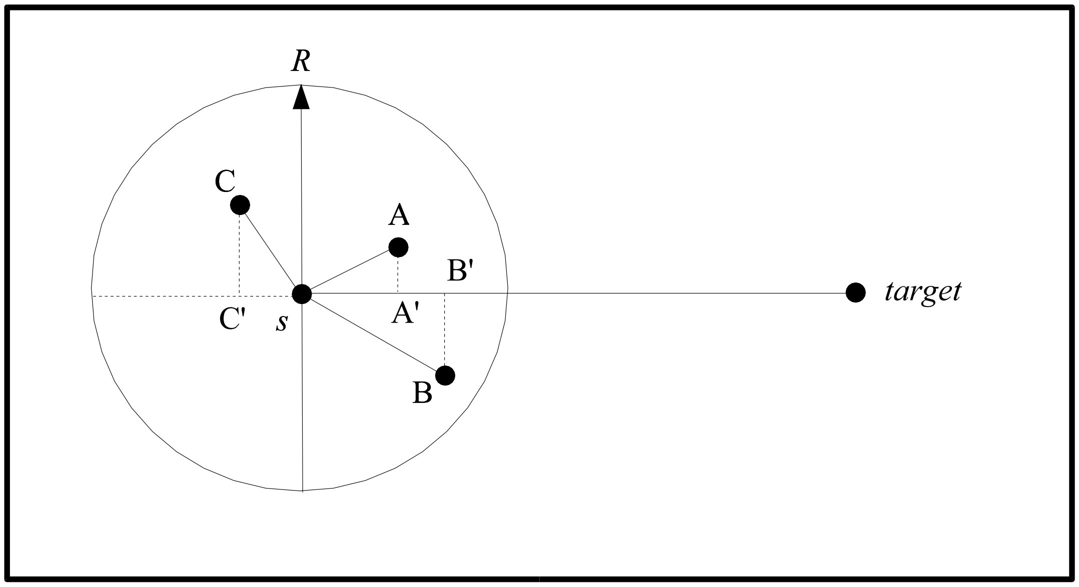

Figure 3.1: Progress: Nodes A and B are in forward direction (positive), C is in backward direction (negative) from source node s.

In the *Random Progress* method [34], at each hop along the route, packets for one node t are randomly forwarded to any neighbouring node that makes forward progress from the current node towards the target. In figure 3.1 node s would randomly choose from nodes A or B.

In the *Most Forward within Radius* method or MFR [41], the neighbour selected is the one which is closest to the destination, that is has the most forward progress.

The MFR algorithm is modified in [15] where nodes can adjust their power to just cover the distance between nodes in communication. This results in forwarding packets for a target to the closest neighbour with forward progress. This is known as *Nearest with Forward Progress* or NFP.

In figure 3.1, we see nodes with forward and backward progress from node s. In MFR, node B would be selected, while with NFP neighbour A would be chosen.

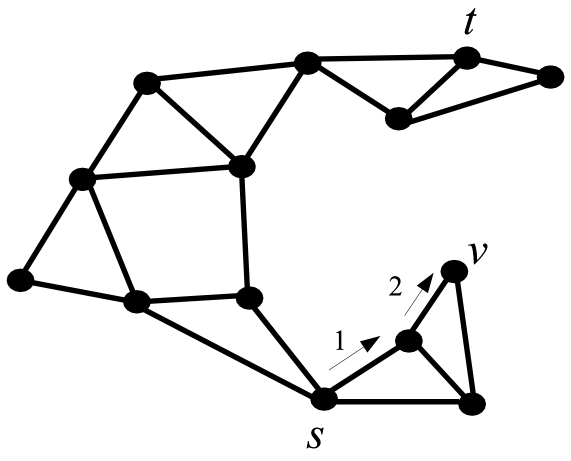

Figure 3.2: Greedy algorithm: Finding a route from source s to destination t fails as node v has no neighbours with forward progress.

Finally, we can see that greedy forwarding algorithms alone cannot guarantee delivery of packets. There are some network topologies, such as the one shown in figure 3.2, where a packet may be forwarded to a node where no forward progress can be made. In these cases, there must be a recovery mechanism. One algorithm that has such an enhancement is *Greedy Perimeter State Routing* which we will see later.

3.4 Compass Routing

In [23, 32] we see the introduction of the notion of *Compass Routing.* The idea comes from the real life problem of a visitor to a city trying to navigate to a local landmark that they can see in the distance. The traveler should take the streets that take them in the direction closest towards the destination.

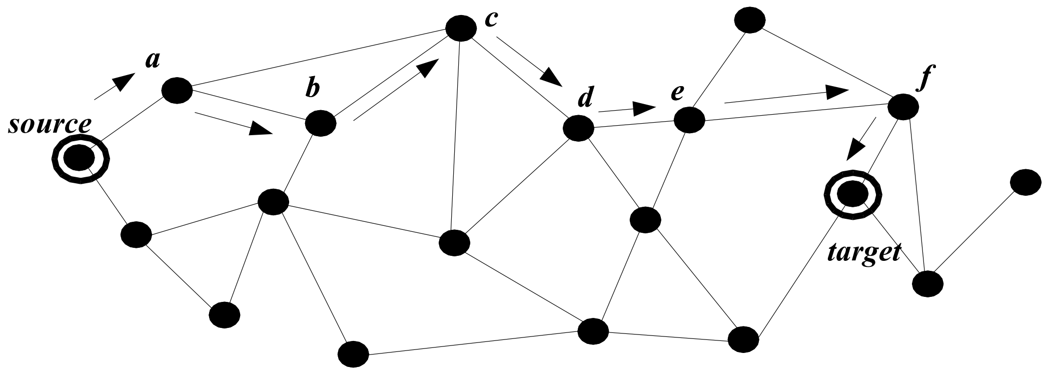

Figure 3.3: Compass Routing from *source* to *target.*

To send a packet from a source node to a destination node, decisions must be made at each node along the route. The next node traversed in the network is chosen based on its direction from the current node in relation to the direction of the destination from the current node. The next node chosen should be the one that has the direction closest to that of the destination. If more than one node fulfills this property (there can be a maximum of two), then one of them is chosen arbitrarily. In figure 3.3 we see the algorithm at work.

Simple compass routing is not sufficient to guarantee delivery of packets from source to target. The author in [23] has shown a network topology which will defeat compass routing.

3.5 Face Routing

In this section, we will describe two *Face Routing* algorithms [6, 23, 32] where packets are progressively forwarded through intermediate nodes across adjacent *faces* of a planar graph towards the destination node.

A connected planar graph will subdivide the plane into a set of adjacent *faces*. These faces are bounded by a polygonal shape made by the edges of the graph. If a packet arrives at a vertex v, which lies on a face *f*, the packet can be forwarded around the boundary of f by using the well established *right-hand rule* [4]. This rule states that if you keep your right hand on the wall while walking forward, all the walls of a maze can be traversed. This rule is the basis for the *face routing* algorithms. Figure 3.4 shows the algorithm developed in [23].

The operation of the *Face-1* algorithm is depicted in figure 3.5. All nodes know their location information along with the location of the target node. The $line(source, target)$ is used to measure progress of the algorithm. The packet is forwarded around the boundary of the face and the point(s) of intersection of the $line(source, target)$ are recorded.

The edge which has the intersection point p closest to the target is chosen as the "flip" edge to the next face. A new $line(p, target)$ is now used as a test for intersection starting at the traversal of next face. This procedure repeats until the target has been reached.

The algorithm will reach the target node with at most $4|m|$ hops where m is the number of edges in the underlying network. At each iteration, the packet is forwarded around the complete boundary of the face. Once that is complete, it must

```
Algorithm Face – 1:

point ← source

repeat

    face is the face of the graph with point on the boundary that intersects
    the line (point, target)

    for each edge (u, v) of face

        if edge (u, v) intersects line (point, target) at a point next and
        dist(next, target) < dist(point, target)

            point ← next

        end if
    end for

    Continue traversing the edges of the face until returning to point

until point == target
```

Figure 3.4: The Face-1 Routing Algorithm.

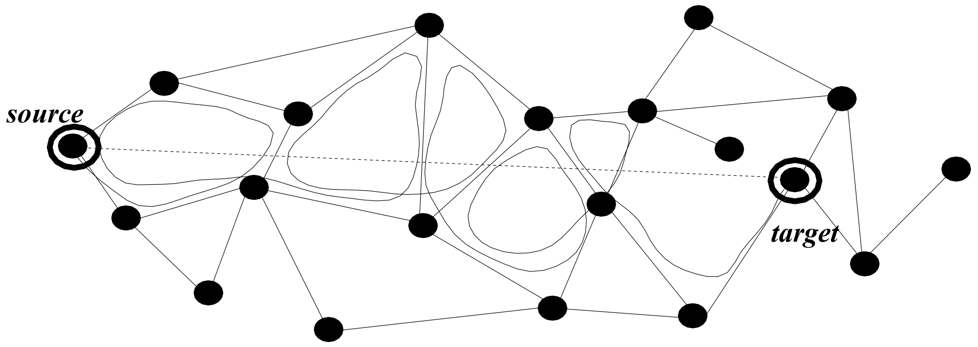

Figure 3.5: Face-1: Routing from *source* to *target*.

be forwarded again, back to the "flip" edge. This can be improved to $3|m|$ hops by forwarding the packet to the next "flip" edge on the shorter of the two paths around the face boundary.

The authors in [6] propose an improvement on the first algorithm. The algorithm *Face-2* is shown in figure 3.6. Instead of traversing the entire boundary of a face at each iteration, the algorithm will "flip" to the next face when an intersection of $line(p, target)$ is found such that this new intersection point is closer to the target than the previous one. This way, the algorithm is ensured to make progress towards the target node.

Algorithm Face – 2:

```
point ← source
repeat
    face is the face of the graph with point on the boundary that intersects
    the line (point, target)

    traverse the face until arriving at an edge (u, v) that intersects
    the line (point, target) at some point  next != point

    point ← next
until point == target
```

Figure 3.6: The Face-2 Routing Algorithm.

In figure 3.7, we can see the operation of the *Face-2* algorithm. The algorithm terminates in a finite number of steps as the distance to the target is shrinking at each iteration. In practice, *Face-2* performs better than *Face-1*, however, it has been shown that in extreme cases it can visit $\Omega(n^2)$ edges of the graph.

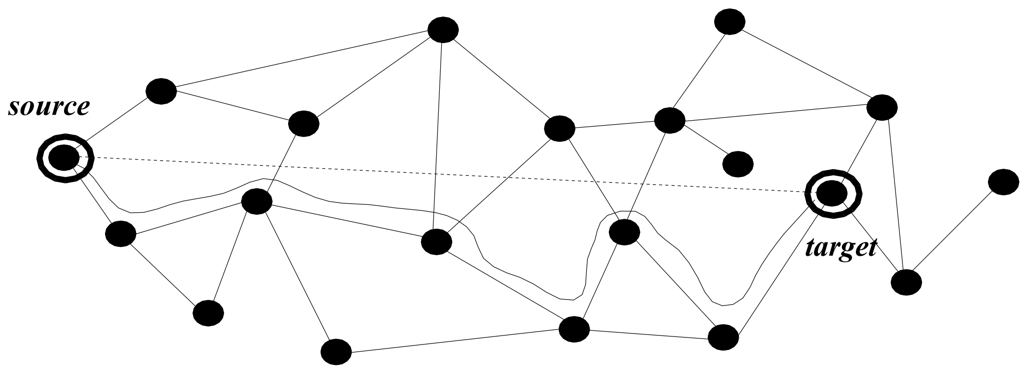

Figure 3.7: Face-2: Routing from *source* to *target*.

3.6 Adaptive Face Routing - AFR

Adaptive Face Routing or *AFR* [24, 26] is an extension of *Face Routing*, which requires an underlying planar network. The algorithm tries to bound the face routing algorithm to minimize the distance traveled from the $line(source, target)$. AFR performs face routing in two steps.

3.6.1 Bounded Face Routing ($BFR[\hat{c_d}]$)

In BFR, the authors assume that an upper-bound $\hat{c_d}$ on the length $c_d(p^*)$ of the shortest route p^* from the source s to the destination t is known beforehand. They let ϵ be an ellipse defined by the locus of all points the sum of whose distance from s and t is $\hat{c_d}$. The ellipse has foci of s and t. By the definition of ϵ, the shortest path from s and t through some point which lies outside ϵ will be longer than $\hat{c_d}$.

The face routing algorithm is now modified in such a way that when traversing faces, we stay within the boundary of the ellipse ϵ. With the definition of BFR we now have the main building block of AFR.

3.6.2 Adaptive Face Routing (AFR)

The problem with BFR is that in a working network, an upper-bound on the length of the shortest path from s to t is not known. The algorithm first must determine an estimate $\tilde{c_d}$ of the unknown value $c_d(p^*)$. To begin, let $\tilde{c_d} := 2\bar{st}$.

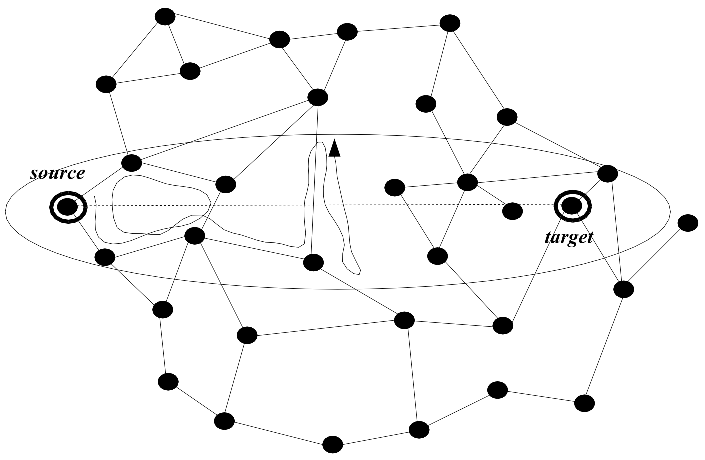

Figure 3.8: Bounded Face Routing (failure, ellipse size chosen too small).

The algorithm continues as follows:

1. Execute BFR[$\tilde{c_d}$].

2. If BFR in 1. succeeded in finding a route to node t then the algorithm is finished. If not, double the estimate of length of shortest path, $\tilde{c_d} := 2\tilde{c_d}$. Repeat process starting at step 1.

Figure 3.8 shows an unsuccessful execution of BFR, the bound was set to small so no path to the target was found. In the following figure (3.9) the bound has been increased, and the algorithm succeeds in finding a path to the target.

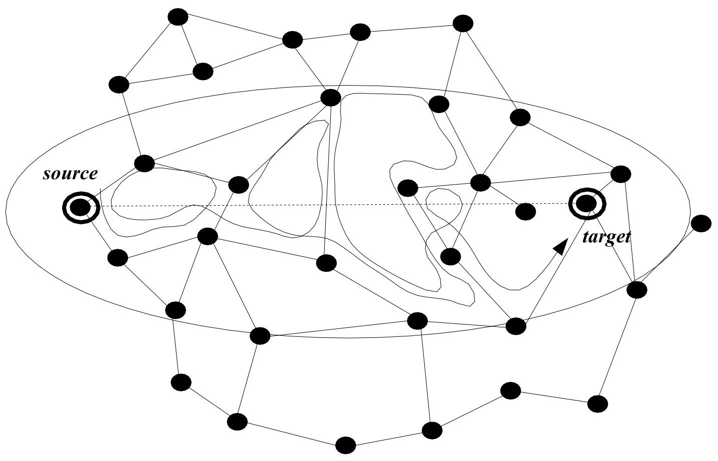

Figure 3.9: Bounded Face Routing (success).

The authors show that the AFR will find a path from s to t with at most $O(c_d^2(p^*))$ hops. They also further expand their AFR algorithm with *Greedy Other Adaptive Face Routing* (GOAFR) [25, 27].

3.7 Greedy Perimeter State Routing

In [19, 20], the authors present *Greedy Perimeter State Routing (GPSR)*. The work is based upon the *Greedy-Face-Greedy (GFG)* algorithm first proposed in [6]. The algorithm uses two methods to forward packets in the network. The first is *greedy forwarding* which is used whenever possible, and the second *perimeter forwarding* is used when the first cannot.

3.7.1 Greedy Forwarding

If nodes know the location of all their neighbours, they can make a locally optimal routing decision by choosing the neighbour which is closest geographically to the eventual destination node. In figure 3.10 we can see an example of the selection of the next hop node. The current node x has just received a message for the *target*. Node x chooses its neighbour y which is closest to the *target*. The circle around node x is the transmission range of node x and the dotted arc is of the circle around the *target* with a radius equal to the distance between it and node y. This process repeats itself until the message makes its way to the *target*.

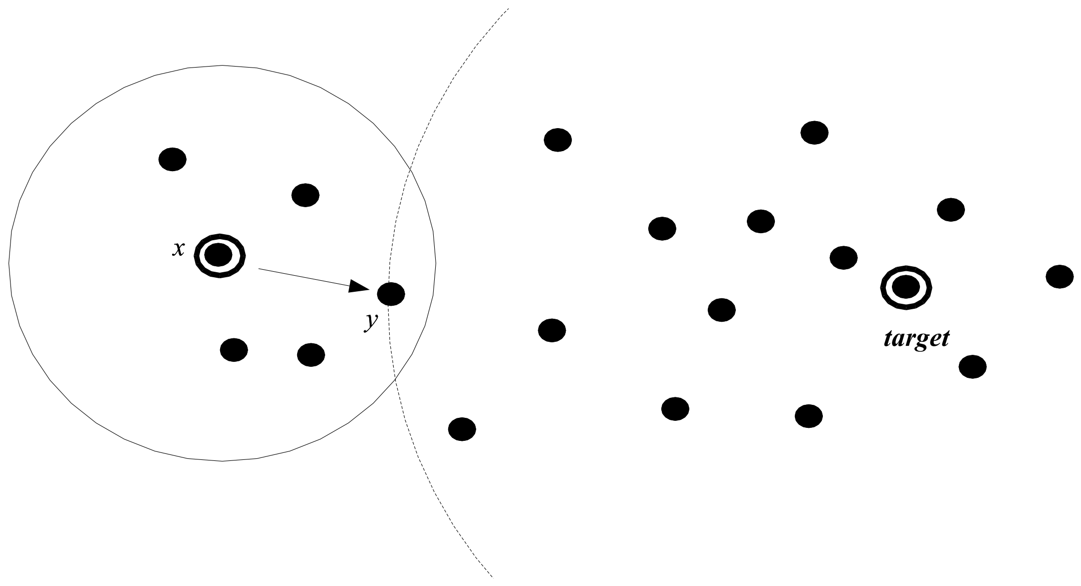

Figure 3.10: GPSR: Greedy Forwarding.

As we have seen earlier, this greedy forwarding may fail to find a route from the *source* to the *destination*. Some network topologies will require that the next hop actually be further away from the destination. To remedy this problem, some recovery mechanism is required.

3.7.2 Perimeter Forwarding

As we can see in figure 3.11, the intersection of the circular transmission range about x and the circle around the *target* with a radius *dist(x, target)* has no neighbours of node x. The shaded region without neighbours is called a *void*. Node x will try to route around the void.

The algorithm now uses the well-known *right-hand rule* to aid in routing around the void. The application of the rule is as follows: when arriving at node x, from y, the next link taken is the next one counterclockwise from edge (x, y). The right hand rule traverses a *face* in a clockwise fashion.

The authors use this cycle-traversing property to route around the *void*. In figure 3.11, traversing $(x \rightarrow a \rightarrow b \rightarrow target \rightarrow d \rightarrow c \rightarrow x)$ using the right-hand rule results in routing around the void (in this case, it included the *target*). This order of node traversal is called the *perimeter*.

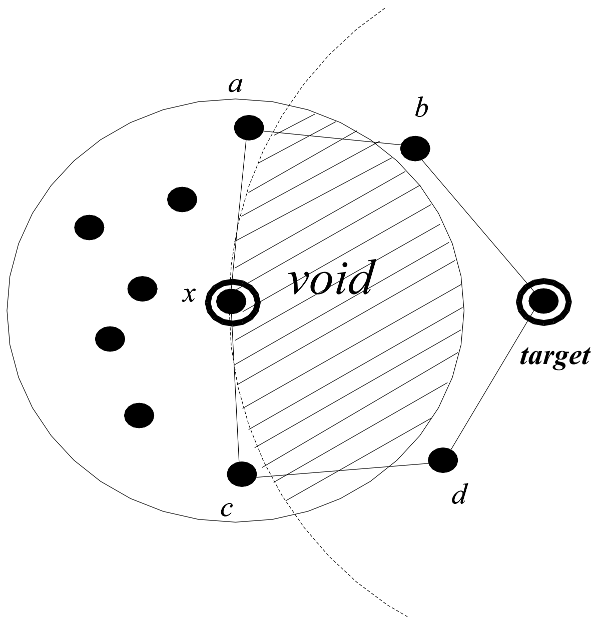

Figure 3.11: GPSR: The *void* and the *perimeter*.

3.7.3 Combining Greedy and Planar Perimeters

The GPSR algorithm combines the greedy forwarding with perimeter forwarding within a *planar* graph when greedy forwarding cannot be achieved. The packets start out in *greedy-mode*, and the source node checks its list of neighbours to see if there is one that is geographically closer to the target node. If such a neighbour is found, the packet is forwarded to this node. If no neighbour is found, the packet is marked and the algorithm switches to *perimeter-mode*.

When in *perimeter-mode*, packets are forwarded using a simple *planar* graph traversal. This works similarly to the face routing algorithm, packets are forwarded across adjacent *faces*.

When the packet switched to perimeter-mode, it was marked with the location of the point where it left greedy-mode. The algorithm uses this information to determine at which point it can return back to greedy-mode and continue. The process repeats itself until the packet reaches the target.

3.8 Location Aided Routing - LAR

Location Aided Routing (LAR) [21, 22, 8] is an on-demand routing algorithm like DSR (which we have seen in the previous chapter). LAR and DREAM (which we will see in the next section) make use of more than simply the mobile nodes location information. They also utilize information on nodes mobility as well, such as a nodes average velocity.

In comparison to DSR, LAR will send its location information within each packet, and in doing so will hopefully reduce the overhead of any subsequent route requests. In DSR, when a node has a packet to send to a destination t, but does not have a route, it will flood the entire network with a route request. LAR, on the other hand makes use of the location information of the mobile nodes to flood within an area in a certain direction. This area is called the *request zone.* By using a request zone, the algorithm hopes to limit the flooding or route request packets to within a portion (hopefully small) of the entire network.

The LAR protocol defines two methods for nodes to use in determining the request zone to which they will forward route request packets.

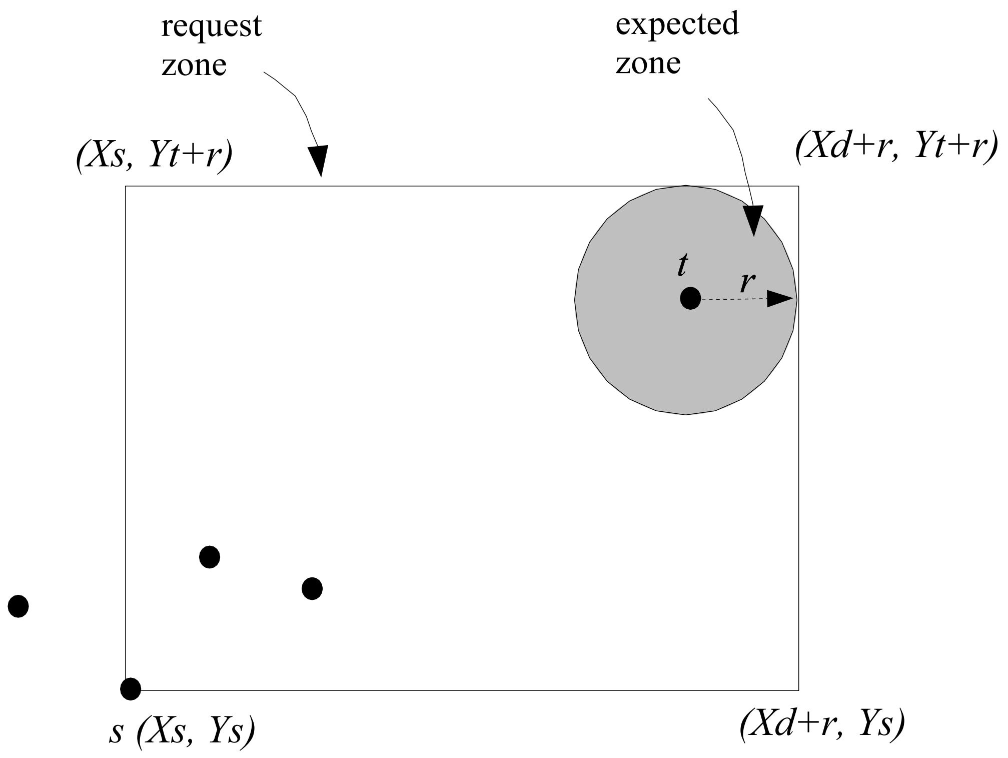

Figure 3.12: LAR Request Zone and Expected Zone: Nodes not inside request zone will not forward packets.

3.8.1 LAR Box

In this method, the request zone is established as a rectangle with the source s in one corner and the *expected zone* in the opposite. The expected zone is a circular area around the destination t which has a radius of the range that node t could have moved since the last time its location was updated. See figure 3.12 to see the request and expected zones in the LAR Box method. Neighbouring nodes which are not in the request zone will not forward the route request packets.

3.8.2 LAR Step

In the LAR Step method, an intermediate node checks to see if it is closer to the destination t than the node which it has just received the packet from. If it determines that it is closer, it will consider itself in the request zone and forward the packet.

3.8.3 Routing

Regardless of whichever of the above procedures are used, the algorithm determines a two stage route discovery mechanism. In the first stage, a packet is forwarded using one of the above two methods. If a *route reply* is not returned within a defined *request timeout* period, the algorithm moves to the second stage. A second route request is then flooded across the entire ad-hoc network (as in DSR). If a route reply is still not received, the current node will mark the destination node t as unreachable. If node t stays unreachable for more that 30 seconds, all packets for t will be dropped.

3.9 Distance Routing Effect Algorithm for Mobility

Unlike Location Aided Routing, the Distance Routing Effect Algorithm for Mobility or *DREAM* algorithm [2, 8] is a *proactive routing protocol.* All nodes keep track of the location information of all nodes in the ad-hoc network. Table maintenance is done by having all nodes sending their location information to all other nodes in the network. This is done smartly, such that this information is sent to nearby nodes at a more frequent rate than that of far away nodes. The premise here is that nodes farther away from you appear to move more slowly, and by changing the rate of updates DREAM hopes to limit the use of bandwidth required for control messages.

To forward a packet, the source, or any intermediate node s calculates the direction of the destination t. It then will use the mobility information it has about t to determine a angular direction *range*. The current packet is forwarded to all of the neighbours of s in this directional range. The area is determined by the two tangents from the current node to a circular range about the destination node t. This circular range is defined as the area that covers all possible locations that node t could have moved since the last time its location was updated and is called the *expected zone*. The area including the two tangents and the *expected zone* is known as the *request zone* as depicted in figure 3.13.

Unlike LAR, which defines the request zone based upon the source and expected zone of the target, the request zone is recalculated at each intermediate node based on the above description. Also, in DREAM these packets are actually *data* packets and not route request packets. LAR is used for route discovery, while DREAM is used to send data packets. When a packet arrives at the destination, an *acknowledgement* or ACK is sent back to the source node using the same mechanism the original data was sent to t.

If for some reason a data packet or the corresponding ACK is lost, the source will move to a *recovery* mechanism. The source node will simply flood the entire network with the packet destined for t.

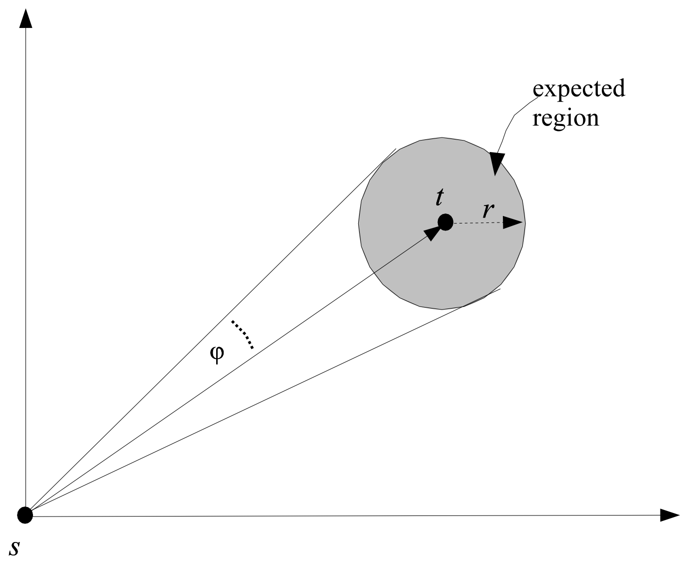

Figure 3.13: DREAM: Request zone and expected region of target.

Chapter 4

Creating a Planar Spanning Subgraph

4.1 Introduction

The positions of nodes in a wireless ad-hoc network are random, and thus the network formed may not be planar. In this chapter we will see various algorithms designed to derive a *planar spanning subgraph* from an underlying network using only local information, that is, using only information about the node and its neighbours location information.

In an ad-hoc network, individual nodes often do not have *global* information about their environment. Nodes will know their location information, and can easily acquire their neighbours location, so local algorithms must be able to operate under these constraints.

As we saw earlier, a planar graph is one where no two edges cross. By a *planar spanning subgraph* we mean that the graph created from these algorithms must be planar, as well as connected.

We will first describe two older algorithms, the *Relative Neighbourhood Graph* and the *Gabriel Graph* which simply check an easily defined region for the presence of any neighbours. Then we will describe a newer algorithm which is more complex, and computes a localized Delaunay triangulation, and removes links to make the graph planar.

4.2 Relative Neighbourhood Graph

This section will describe the production of the *Relative Neighbourhood Graph (RNG)* [42]. A planar spanning subgraph of a network can be obtained by applying a test to every pair of nodes which are within the transmission range of each other. Let A and B be two nodes whose distance is less than the transmission range R of the network. Consider the region delimited by the intersection of the circles centered at A and B, respectively, where the radius of the circles is determined by the transmission of the stations at A and B, see Figure 4.1. If there is no node in the region then the link between A and B is kept. If however there is a node C in the region depicted in Figure 4.1, then nodes A and B remove their direct link.

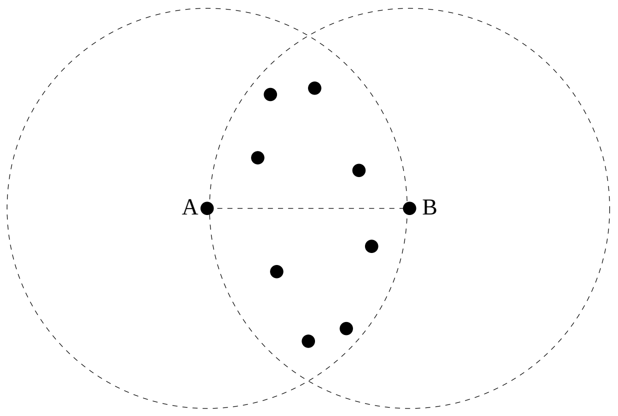

Figure 4.1: The *RNG* test for producing a planar spanning subgraph.

The resulting *RNG* graph suffers from the *multiple hop effect* because the elimination of crossings is done by elimination of longer links.

4.3 Gabriel Test

One of the most important tests for eliminating crossings in a wireless network is called the Gabriel test [11] which, similarly to the *RNG* test, is applied to every link of the network. The main difference between the Gabriel test and the previous test is that it considers a smaller size region for link elimination.

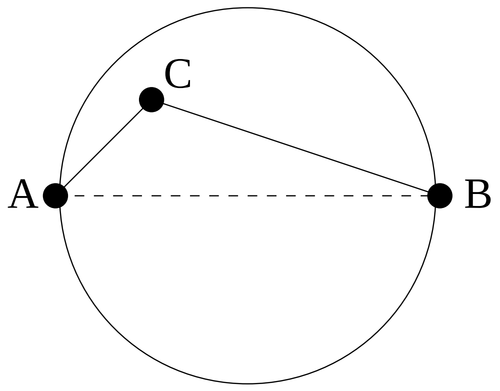

Figure 4.2: Eliminating an unnecessary link (dashed line AB) with the Gabriel test.

Let A and B be two nodes whose distance is less than the transmission range R of the network. In the Gabriel test, if there is no node in the circle with diameter AB then the link between A and B is kept. If however there is a node C in the circle with diameter AB, as depicted in Figure 4.2, then nodes A and B remove their direct link. In particular, when A (respectively, B) is queried on routing data to B (respectively, A) the routing table at A (respectively, B) forwards the data through C (or some other similar node if more than one node is in the circle with diameter AB.

The advantage of doing this rerouting of data is that the resulting graph is a planar spanning subgraph on which we can apply planar routing algorithms such as *Face Routing* or *GPSR* for discovering a route from source to target.

In comparison with the RNG test, the Gabriel test shrinks the test region and creates a planar spanning subgraph that keeps some of the links that would be eliminated by the *RNG* test. However, the resulting graph of the Gabriel test still may suffer from the multiple hop effect.

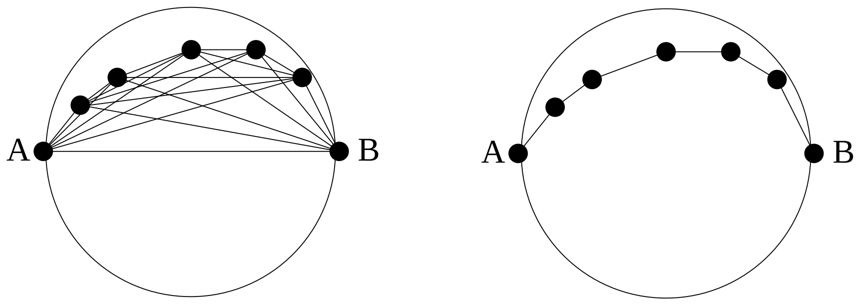

Figure 4.3: Multiple hop effect when eliminating a link (line segment AB) via the Gabriel test.

For example, consider a set of nodes as depicted in the left-hand side of Figure 4.3. All nodes are mutually reachable. However, when we apply the Gabriel test the configuration in the right-hand side of Figure 4.3 results. We can see that although nodes A and B could have reached each other directly in a single hop instead they must direct their data through a sequence of many hops.

4.4 Localized Unit Delaunay Triangulation

In a recent work, the authors in [29] describe an algorithm to locally construct a planar spanning subgraph of the unit disk graph. They use the construction of $LDel^{(1)}(V)$ as the basis of the algorithm. The authors show that the $LDel^{(1)}(V)$ may not be planar, so they must make it planar by removing links. The result of their algorithm is $PLDel(V)$ and they show that the Unit Delaunay triangulation, $UDel(V)$, is a subgraph of their result.

Note that a triangle $\triangle uvw$ belongs to the Delaunay triangulation $Del(V)$ if its circumcircle $disk(u,v,w)$ does not contain any other node of V in its interior. The algorithm is as follows:

Algorithm: Localized Unit Delaunay Triangulation

1. All nodes broadcast an *announce* message with its identity and location to its neighbours $N(u)$ (beacon) as well as listens for messages from its neighbours.

2. Node u has received all the information about its neighbours and then computes the Delaunay triangulation $Del(N(u))$ of u and its 1-hop neighbours.

3. For every $edge(u, v)$ of $Del(N(u))$, let $\triangle uvw$ and $\triangle uvz$, be two triangles incident on $edge(u, v)$. The edge(u,v) is a Gabriel edge if both angles $\angle uwv$ and $\angle uzv$ are less than $\pi/2$. The node u marks all Gabriel edges and these will never be removed.

4. For all nodes u, they find all triangles $\triangle uvw$ of $Del(N(u))$ such that all edges of $\triangle uvw$ have length of ≤ 1 unit and $\angle wuv \geq \pi/3$. The node sorts the edges of triangles in clockwise order and broadcasts a *proposal* message (includes u's ID, followed by the IDs of node v and a bit to tell neighbours if rule is met). Then node u listens for messages from its neighbours.

5. For all nodes u that receives a *proposal* message from v, will check if v if proposing to add $\triangle uvw$ and $\triangle uvz$ to $LDel^{(1)}(V)$. Node u will then check if any other triangles proposed by other nodes is part of $Del(N(u))$, and then broadcasts an *accept* message (containing IDs and information on adjacent vertices which form a triangle belonging to $Del(N(u))$).

6. Node u will add edges uv and uw if the triangle $\triangle uvw$ is in $Del(N(u))$ and both v and w have send a propose or accept message for said triangle.

7. Now, to being making the subgraph planar, node u will broadcast a *check* message (its ID and list of vertices v such that $uv \in LDel^{(1)}(V)$)

8. Node u, using check messages from its neighbours, checks for all 1-local Delaunay triangles $\triangle uvw$ to see if there is a node inside the circumcircle. If found, then u will remove the triangle. Node u then sorts the remaining edges (Gabriel edges and those that were not just discarded) in clockwise fashion.

9. Then all nodes u will send an *alive* message (including its ID and IDs of neighbouring vertices and a bit (between nodes v and w) on whether triangle $\triangle uvw$ was kept in $LDel^{(1)}(V)$).

10. Node u will keep edge uv if it is a Gabriel edge or it is part of a triangle $\triangle uvw$ that was kept in alive messages from all nodes u, v, and w.

11. The algorithm is complete and the planar subgraph $PLDel(V)$ has been created.

The authors further their work in [43] where they compute a planar spanner with bounded degree vertices using the Yao [45] structure on the algorithm defined here. They also describe an algorithm in [30] for producing Delaunay triangulations over the underlying graph in Bluetooth scatternet formation.

Chapter 5

The Morelia Test

5.1 Introduction

In this chapter we present a new localized algorithm for creating a planar spanning subgraph called the *Morelia Test* [5]. The goal is to improve the performance of routing with the Morelia graph in comparison to the Gabriel graph. The test produces a subgraph that keeps longer links on average than the Gabriel graph and helps reduce the impact of the *multi-hop effect.* We simulate the face routing [6, 23] algorithm on both the *Gabriel graph* and the *Morelia graph* to show that routing on the Morelia graph takes fewer hops on average, especially when the network has areas which are sparse.

Next, we will describe the goals of the Morelia Test, and then define how the algorithm operates. We will then discuss the overhead of the Morelia Test and how to improve the operation when the network is very dense. Finally, we discuss the series of simulations that were designed and implemented to show the following two main objectives:

1. The Morelia Graph has longer links on average than the Gabriel Graph

2. The Face Routing algorithm performs with a fewer number of hops on average on the Morelia Graph than the Gabriel Graph.

5.2 Goals

There are two important goals we should be concerned in our reduction from the original wireless network to the geometric planar graph:

1. **Keep long links.** This is required so that we can prevent unnecessary large number of hops from source to destination that require extra processing at the nodes and may introduce failures.

2. **Eliminate crossings.** This is required in order to create a planar underlying graph. Furthermore, the method employed must be efficient and be based on local tests and the resulting graph must be a spanner of the original network.

The first goal described above may not necessarily be consistent with minimizing power consumption. Our hope is that the new subgraph created will result in fewer hops for routing messages. Consider the configuration depicted in Figure 4.2. If node C is inside the circle determined by the diameter AB then an elementary calculation shows that the power required to reach directly from A to B is less than the power required to reach from A to B via node C. However, multi-hopping may cause extra overhead due to local processing, congestion, or an increased probability of a lost message.

Concerning the second goal above, tests must be applied locally by the individual nodes, since in many cases there is no global information available about the network and the network can keep on changing. Thus, a node will decide to keep or delete an existing edge solely based on the requirement for eliminating crossings. As a result, long links that typically are involved in many crossings may be deleted and the resulting routes will have an unnecessarily large number of hops from source to destination.

A side effect of these goals is that we aim to improved or reduce the number of hops involved in the face routing algorithm.

5.3 Morelia Test Defined

The precise specification of the Morelia applied to a link AB is given below (refer to Figure 5.1)

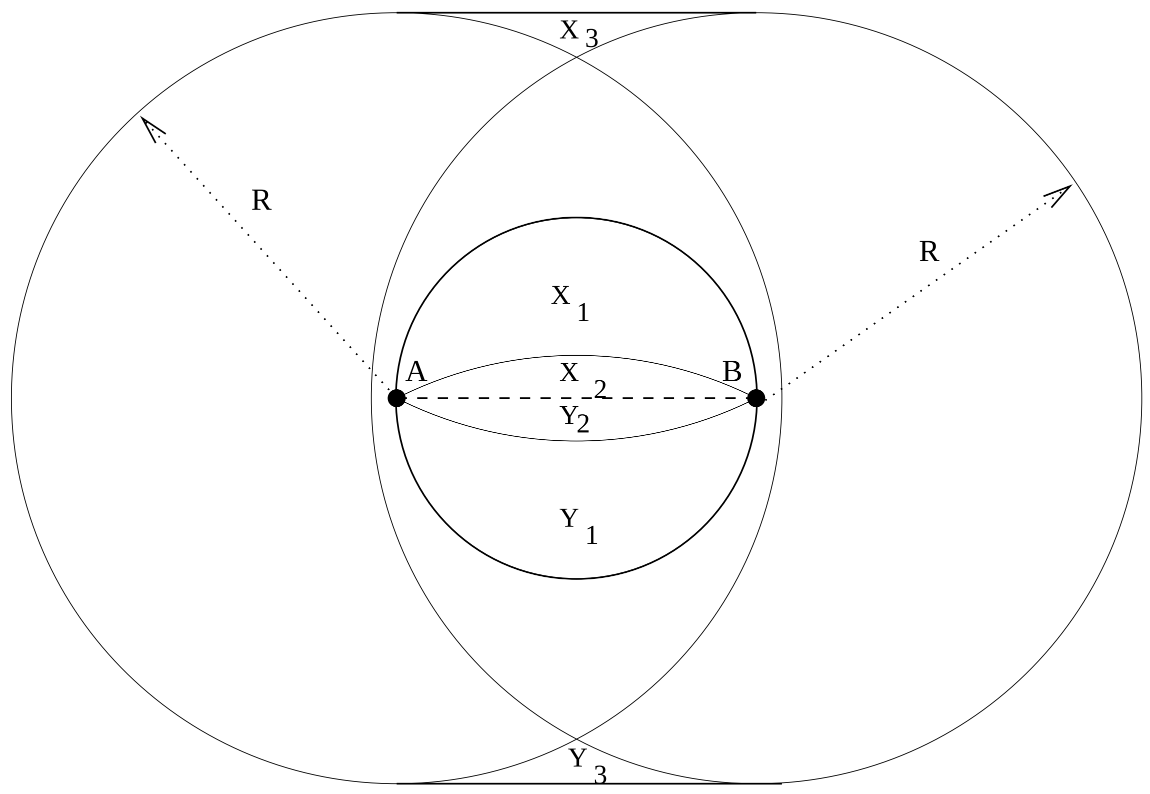

Figure 5.1: Morelia Test Defined.

The regions examined are defined as X_1, X_2, X_3, Y_1, Y_2 and Y_3. We also have the inner circle, which is the same as the one used in the Gabriel test, and the two circles corresponding to the transmission radius, R, of A and B. The inner circle is divided into regions X_1 and X_2. The region X_2 is defined by the arc segment with a radius R from the lower intersection point of the two transmission radii. Region X_1 is the region in the inner circle above this arc. The regions Y_1 and Y_2 are defined in a similar manner as X_1 and X_2. Regions X_3 and Y_3 are regions which cannot be reached by either A or B and are define by the intersection point of the two transmission radii, and the points on the circles (directly above for X_3, below for Y_3, A and B) which forms a perpendicular line with line AB.

Morelia Test Rules.

1. If there is at least one node in $X_1 \cup X_2$ and at least one node in $Y_1 \cup Y_2$ then the link AB is removed.

2. If there is at least one node in X_1 and no node in $X_2 \cup Y_1 \cup Y_2$ then the node A checks whether any node in $N(A)$ creates a link with nodes in X_1 that crosses AB. If such a crossing occurs, link AB is removed an A sends a message to B to remove the link as well. Similarly node B performs a check of nodes in $N(B)$ for a crossing of the link AB and informs node A if a crossing is found and AB is to be removed.

3. If there is at least one node in Y_1 and no node in $Y_2 \cup X_1 \cup X_2$ (symmetric to Rule 2) then the node A checks whether any node in $N(A)$ creates a link with nodes in Y_1 that crosses AB. If such a crossing occurs, link AB is removed an A sends a message to B to remove the link as well. Similarly node B performs a check of nodes in $N(B)$ for a crossing of the link AB and informs node A if a crossing is found and AB is to be removed.

4. If there is at least one node in X_2 and no node in $Y_1 \cup Y_2$ then the node A inspects the nodes in $N(A)$ to check whether any node there creates a link with

nodes in $X_1 \cup X_2$ that crosses AB. If such a crossing occurs, link AB is removed and A sends a message to B to remove the link as well. Furthermore A sends a message to nodes in X_2 with a request to send back information whether there is a node in the region Y_3. If A receives a message that a node exits in Y_3 then AB is removed and node B is informed to remove the link as well.

5. If there is at least one node in Y_2 and no node in $X_1 \cup X_2$ (symmetric to Rule 4), the node A inspects the nodes in $N(A)$ for a possible crossing of AB. If such a crossing occurs, link AB is removed and A sends a message to nodes in Y_2 with a request to send back information whether there is a node in the region X_3. If A receives a message that a node exits in X_3 then AB is removed and node B is informed to remove the link as well.

The Figure 5.2 illustrates how, unlike the Gabriel test, the Morelia test will check for crossings prior to eliminating a link. It will eliminate link AB because it detects crossings, but it will not eliminate it when no crossing is seen.

Notice that in the rule 1 of the Morelia test, nodes A and B look only at nodes that are in $N(A)$ and $N(B)$ respectively, i.e., nodes that are one hop away. In rules 2, 3, 4, and 5 node A checks nodes in $N(A)$, node B checks nodes in $N(B)$ and both A and B possibly check $N(x)$ for nodes x in $\in X_2 \cup Y_2$. Thus A or B are checking nodes that are at most two hops away. Therefore, the Morelia test is a local test.

Theorem 1. *If network N is connected then the application of the Morelia test to all links of N produces network N' which is a connected planar spanning subgraph of N. Furthermore, N' contains the Gabriel graph of N as its subgraph.*

Proof. Every edge in N that is kept by the Gabriel test is also kept by the Morelia test. Thus, N' contains the Gabriel graph of N as its subgraph. Since the Gabriel test produces a connected subgraph of N, the subnetwork N' is connected.

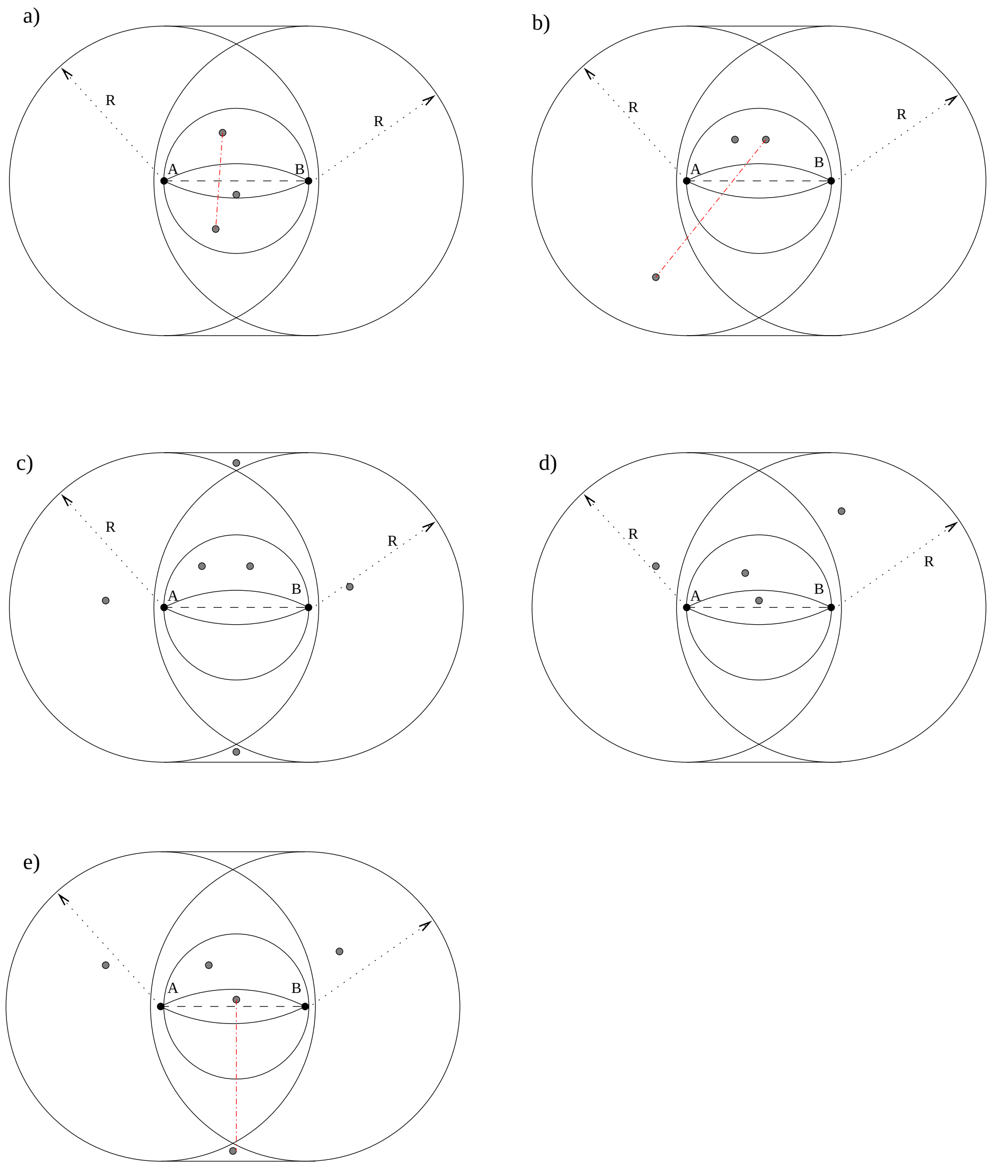

Figure 5.2: Examples of the Morelia test situations: a) AB deleted by rule 1, b) AB deleted by rule 2, c) AB kept, d) AB kept, e) AB deleted by rule 4.

Assume that there is a link e in N' that crosses a link AB of N'. Since the length of any link in N is at most R, both ends of e are in $N(A) \cup N(B) \cup X_3 \cup Y_3$. If one of ends of e is in the circle with diameter AB then AB would be deleted by the Morelia test. If both ends of e are outside the circle with diameter AB then one of the ends of the edge AB must be inside the circle with diameter e, since the edges crosses each other. However, the Morelia test applied to e would eliminate e because of it being crossed by AB. Thus there can be no crossing in N'. □

In Figure 5.3 we show how the Morelia test keeps some of the long links. The left-hand side shows a link AB and the nodes inside the circle with the diameter AB and the right-hand side shows what edges are kept by the Morelia test.

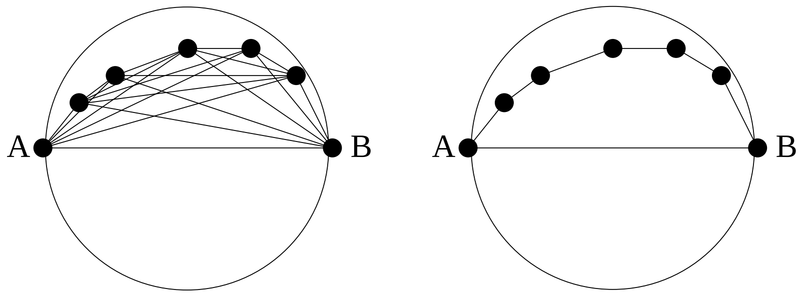

Figure 5.3: Multiple hop elimination (line segment AB) via the Morelia test.

5.4 Overhead of the Morelia Test

In this section we will discuss the overhead of the Morelia Test. As the test is performed for every link in the network, it is possible that the Morelia test will send messages on every link. In a complete network, where all nodes are in communication, the number of links is O(n^2). In a dense network, the Morelia test may introduce O(n^2) messages. We will revisit this situation later.

In Rule 1 of the Morelia test node A or B needs to determine what nodes are in the circle with AB as its diameter and the complexity of it is not much greater than that of the corresponding Gabriel test.

Rules 2 and 3 for A involves nodes that are in $N(A)$ and each node of the network needs to know its neighbours anyway. To check for crossings of two line segments involves simple geometrical computation of complexity $O(1)$, and the same applies to node B. The exchange of messages between A and B confirming deletion or retention of the edge is also simple.

Probability of sending a message to other node to remove link AB.

In rules 2 and 3, node A will only send a message to B to remove the link AB if the node that creates a link with a node in X_1 is a $N(A)$ and not a $N(B)$ (the reverse is true for B sending a message to remove the link to A). Similarly, for rules 4 and 5, a message will only be sent to the other node to remove link AB if the culprit node is not a neighbour of both A and B.

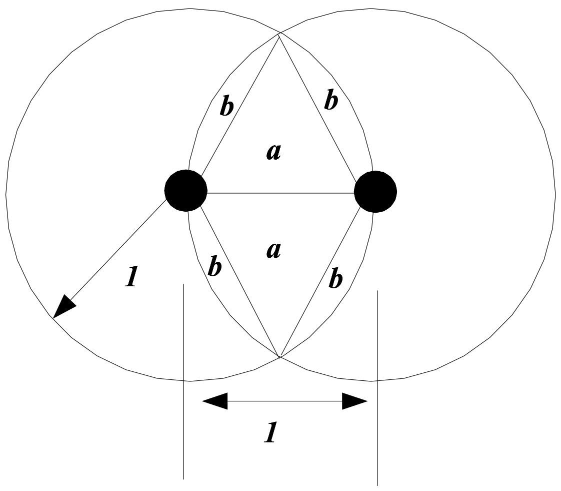

Figure 5.4: Computing area of circle overlap

The probability that a node which is a neighbour of A is a neighbour of B as well is the percentage area of the overlap to the total area of two transmission ranges.

To find the area of the overlapping region of a circle when nodes are at distance

R (same as the transmission range) we can treat this *worst case* scenario as a unit circle problem (see figure 5.4).

First, we draw two unit equilateral triangles, a, inside the overlapping area, with their vertices at the nodes and the intersections of the circular transmission range. Next, we find the area of each equilateral triangle to be:

$$Area(a) = \sqrt{3}/4 \tag{5.1}$$

We must then find the area of the four lens-shaped pieces, b, outside the equilateral triangles. The area of each one of these lens shapes is the difference between 1/6 of a unit circle and the area of the unit equilateral triangle a. Each lens has an area of:

$$Area(b) = \pi/6 - \sqrt{3}/4 \tag{5.2}$$

Then, add up the pieces:

$$Area(overlap) = 2 \cdot Area(a) + 4 \cdot Area(b) \tag{5.3}$$

The total area is equal to twice the area of a circle defined by a transmission range, minus the area of the circle overlap.

$$Area(total) = 2\pi - Area(overlap) \tag{5.4}$$

Now we can calculate the probability that a node will be both a neighbour of A and B. The probability is:

$$p = \frac{Area(overlap)}{Area(total)} \tag{5.5}$$

When we work this probability out based on the unit circle, we see that $p \approx 0.24$. That means that in the worst case, if nodes are uniformly randomly distributed, approximately 24% of the links will have a random node in common. In terms of sending messages, the probability of A sending a message to B (or B to A) will be $1-p$ or approximately 76%.

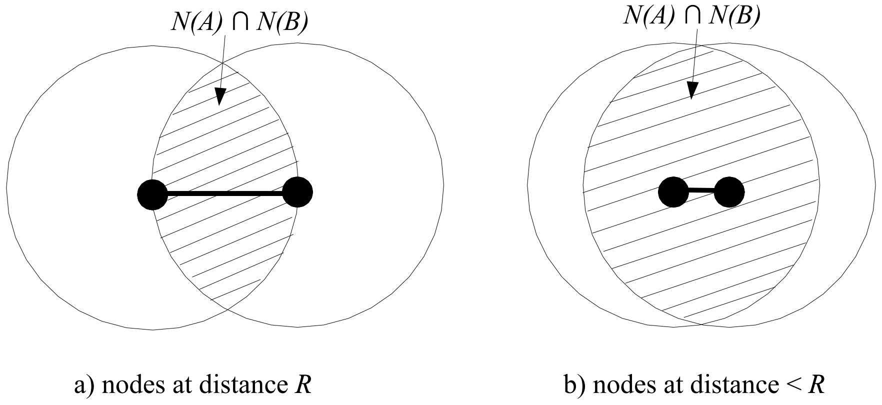

Figure 5.5: Area of circle intersection ($N(A) \cap N(B)$) increases as nodes move closer together.

Figure 5.5 shows that likelihood of this message being sent from A to B (or from B to A) is less likely the closer together A and B are to each other. And in our simulations, we see that as the network becomes more dense, the number of this type of message is reduced as the area of overlap increases.

Existence of Nodes in Regions X_3 and Y_3

Rules 4 and 5 of the test involves the existence of nodes in X_3 and Y_3 that are outside of $N(A) \cup N(B)$. However, all that is needed for A or B is to send a message to the nodes in region X_2 or Y_2 asking the question "is there any node in Y_3 or X_3?" respectively. The region X_3 or Y_3 can be specified by the three corners of the region. Thus, although these rules involve nodes that are two hops away from A and B, it does not create a significant overhead or delay. We show now that the size of the region X_2 or Y_2 is a smaller part of the circle with diameter AB. Since Rule 4 and 5 are used only when X_2 or Y_2 is the only region of the circle containing a node of the network, the probability of using these rules is also smaller.

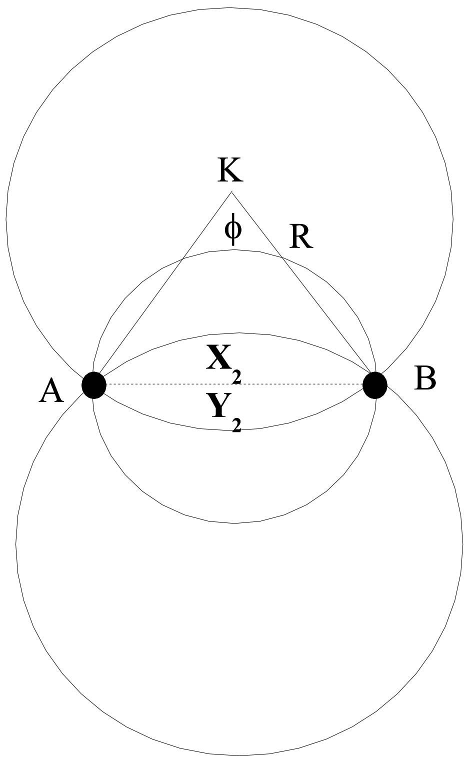

Figure 5.6: The arc segment X_2 (and its symmetric one Y_2) used in Rule 4.

We can easily calculate the upper bound on the ratio of $|X_2|$, the area of X_2 to $|X_1|$, the area of X_1 to get an indication on how often Rules 4 and 5 are used in comparison to the other rules (see figure 5.6).

It is easy to see that the ratio $|X_2|/|X_1|$ is getting smaller when the angle ϕ is getting smaller. Thus we get an upper bound on $|X_2|/|X_1|$ by setting $\phi = \pi/3$. In this case the length of AB equals R and the area of the circle determined by the diameter AB is exactly $\pi R^2/4$, and the area of the circular sector X_1 is equal to $\pi R^2 - \sqrt(3) R^2/4 = R^2(\frac{2\pi - 3\sqrt(3)}{12})$. Thus $\frac{|X_2|}{|X_1|} = \frac{2\pi - 3\sqrt(3)/12}{\pi R^2/8 - 2\pi - 3\sqrt(3)/12} = \frac{4\pi - 6\sqrt(3)}{3\pi - 4\pi - 6\sqrt(3)} < 0.3$.

Thus if a node is located in the circle with diameter AB and its placement is random, it will be in the area $X_2 \cup Y_2$ with probability less than 0.3.

5.5 Handling a Dense Topology

If a network is dense then the Morelia test may eliminate many longer links and keep the shorter ones. In this case we can prevent the elimination of longer links by selecting in network N some of its nodes, called *leaders* that would be produce a sparser subnetwork N'. The selection of leaders is done so that selection of nodes for N' is done by a local *leader election*, and any node of N is at distance less than R from a node in N'. Furthermore, N' must be connected. If N' satisfies these properties then u sending a message to y can be done as follows: u sends the message to a node u' in N' in its neighbourhood, which forwards the message in network N' to a node v' that contains v in its neighbourhood and v' finally sends the message to v.

In this section we propose a leader election algorithm that generates N' satisfying the conditions above. We have a parameter $r \leq \sqrt{R}$ that determines how sparse the network N' is. The main parameters used in the leader election process are the coordinates of the nodes. Informally, we can picture the plane divided into squares by horizontal lines through points $[0,0], [r,0], \ldots$ and vertical lines through points $[0,0], [0,r], \ldots$. If a network N contains a node in a square then at least one node of N is elected or designated as a leader. Notice that two nodes u and v with coordinates $[u_1, u_2]$ and $[v_1, v_2]$ are in the same square if $\lfloor u_1/r \rfloor = \lfloor v_1/r \rfloor$ and $\lfloor u_2/r \rfloor = \lfloor v_2/r \rfloor$. Since $r \leq \sqrt{R}$, all nodes within a square are within transmission range of each other which makes the leader election simpler. For each node the label used in the leader election process will be the sum of the coordinates of the node.

Leader Election.

1. Each node broadcasts its coordinates (x, y).

2. Each node selects the node having the maximum label selected among its own and the ones it received from all nodes *in the same square* and calls it the *primary leader* of the square.

3. Each primary leader of a square calculates the list L_1 of squares that contain nodes within its transmission range. For each square in L_1 the leader selects the node in that square having the highest label among those it can communicate with and designates it a secondary leader of the square.

4. Each leader of a square broadcasts a message to all nodes in its own square asking them to report to it which neighbouring squares they can communicate with. From the reported squares the leader forms a list L_2 of squares that are not in L_1.

5. For every square s in L_2 the leader designates the node x in its own square that communicates with a node y in s having the highest label to be the tertiary leader in the square. If several nodes have this property than the node with highest label is selected. Furthermore p_1 asks x to send a message to y to inform y that y is to become a tertiary leader. Node y informs its primary leader about it.

6. The primary leader informs all leaders in its squares leader

Network $N'(r)$ consists of all the primary, secondary, and tertiary leaders. Notice that the election is local and no message that needs more that 3 hops is needed.

Lemma 1. *Let s_1 and s_2 be two squares with leaders p_1 and p_2 such that there is a link in N between two nodes in these squares. Then there is a path of length at most 3 between p_1 and p_2 in $N'(r)$.*

Proof As far as the network $N'(r)$ is concerned, the following cases can arise:

a) If the leaders p_1 and p_2 are within the transmission range R of each other then p_1 and p_2 select each other as secondary leaders and no new leaders are added. There is an edge between p_1 and p_2 in $N'(r)$.

b) If the leaders p_1 and p_2 are not within transmission range R of each other, p_1 cannot communicate with any node in s_2 but p_2 can communicate with a node in s_1 then the node selected by p_1 as the tertiary leader is the same as the node selected by p_2 as the secondary leader and there is a path of length 2 between p_1 and p_2 in $N'(r)$.

c) If the leaders p_1 and p_2 are not within transmission range, p_1 cannot communicate with any node in s_2, and p_2 cannot communicate with any node in s_1 then there is a path between p_1 and p_2 of length 3 in $N'(r)$ that contains one tertiary leader in s_1 and one tertiary leader in s_2. $\square$

Theorem 2. *If network N is connected then the network $N'(r)$ of the leaders produces by the leader election algorithm above is a connected network such that any node of N is at distance at most R from a node in $N'(r)$.*

Proof. Let p and p' be two nodes of $N'(r)$. Let P be a path in N connecting nodes p and p'. Let $p_1 = p, p_2, \ldots, p_n = p'$ be the sequence of leaders of squares corresponding to the squares the path P is traversing on the way from u to v. By lemma 1, there is a path between p_i and p_{i+1} in $N'(r)$ and thus there is a path in $N'(r)$ between p and p'. Thus, $N'(r)$ is connected.

Any square of the plane that contains a node of N contains a primary leader. Since $r \leq \sqrt{R}$, any node in a square is at distance at most R from the primary leader of the square, a node in $N'(r)$. $\square$

Clearly changing the parameter r can change the resulting network N'. This needs to be studied experimentally.

5.6 Simulations and Case Study

In this section we present our simulations designed to test the Morelia Test in comparison with the Gabriel Test. The purpose of the simulations was to give an indication on how our algorithm actually performs in the real world. We study the operation and performance of our algorithm on randomly generated graphs.

5.6.1 Goals

There were two main goals to achieve with our simulations.

1. To compare the average link length in the planar spanning subgraph created using the Morelia and Gabriel tests.
2. To compare the number of hops required to deliver messages with face routing on the planar spanning subgraph created by the Morelia and Gabriel tests.

5.6.2 Network Model

A simulation was designed that allowed the creation of a random network. A square area (called a *grid*) could be defined, nodes could be created with a specific transmission range and placed at random coordinates on the grid. We use the UDG model, where all nodes have the same transmission range. Each network simulation consisted of 50 nodes each with a transmission range of 250 units. The test area was a square grid on which the 50 nodes were randomly placed. This setup was chosen as it has been seen in other simulations such as those performed by [8]. This network setup also allowed us to test the performance with different node densities by simply varying the grid size.

5.6.3 Analysis of Morelia Graph vs. Gabriel Graph

The Gabriel Test and the Morelia Test were run on the *same* series of random graphs. The following are the main metrics that were traced in the simulation:

1. Average length of Links on each planar spanning subgraph.
2. Number of links in each planar spanning subgraph.
3. Number of Messages introduced by Morelia Test (messages sent to neighbour to remove current link, and messages sent to nodes in region X_2 or Y_2).
4. Number of nodes in Gabriel circle when Morelia keeps a link

Scenario 1 - Uniform Random Graph

The first test, Test 1, was to generate a series of random networks. The grid size was the only parameter what was altered for a series of trials. This had the effect of increasing or decreasing the network density (the smaller the grid size, the the closer together the nodes, the greater number of links, the higher degree vertices).

Figure 5.7 shows the average link length of the Gabriel and Morelia graphs. The Morelia graph indeed does keep slightly longer links on average. The average increase in length was 4.86 units or 5.76%.

Figure 5.8 shows the average number of links kept in the planar spanning subgraph for both tests. The Morelia test keeps more links on average in the spanner than the Gabriel test. The average increase in links kept in the Morelia graph was 6.07 or 7.49% over the Gabriel graph.

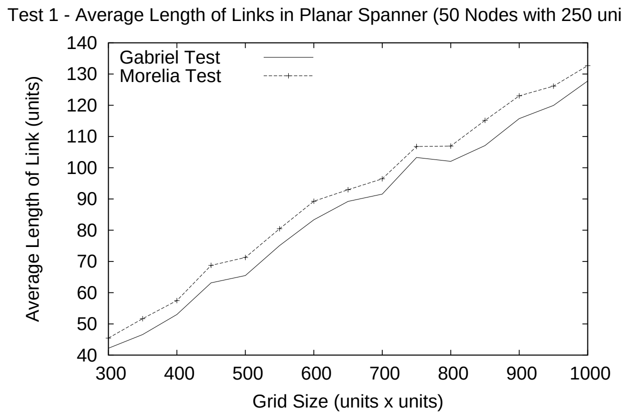

Figure 5.7: Test 1 - Average link length in planar spanning subgraph with Gabriel and Morelia Tests.

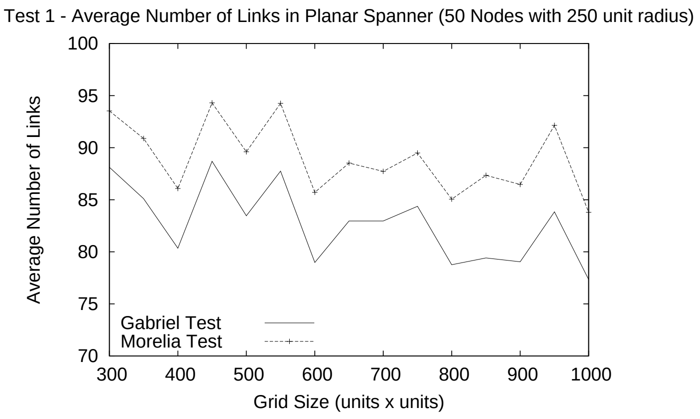

Figure 5.8: Test 1 - Average number of links in planar spanning subgraph with Gabriel and Morelia Tests.

The results in figure 5.9 show the messages introduced in the network by the Morelia test. The upper line shows the total number of messages, the middle line shows the portion of the number of messages sent to nodes in region X_2 or Y_2 to check for presence of nodes in region X_3 or Y_3, and the bottom line shows the portion of the messages sent from node A to B (or vice versa) to inform of a link removal. Notice, for messages sent from A to B (or vice versa) to remove link AB, that the

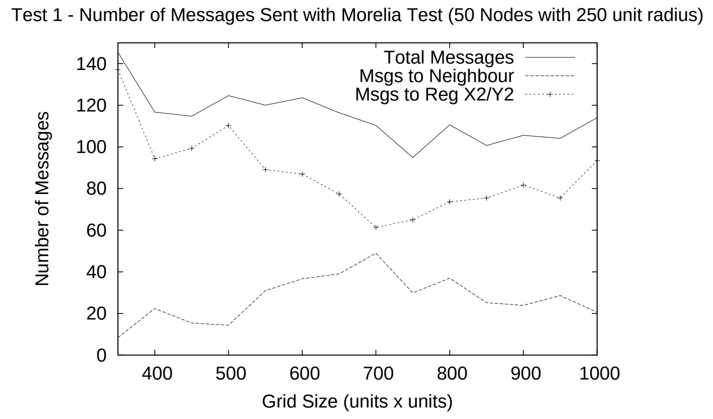

Figure 5.9: Test 1 - Number of Messages Introduced by Morelia Test.

number of messages sent decreases as the network density increases.

We also kept track of the number of nodes inside the Gabriel test circle for links removed from the Gabriel graph, but remaining after the Morelia test. This would give an indication to approximately the number of hops that could be saved by keeping the Morelia link. As expected, the number of nodes inside the region increased with the density of the network. The numbers recorded ranged from an average of 3.5 nodes in our dense topology to 1.3 nodes in our sparse topology.

Scenario 2 - Random Graph with a Sparse Region

This simulation, Test 2, was designed to test the planar spanning subgraph creation when there exists an area of the graph which has a less dense topology. A region 500x500 units within the test grid was defined to have a node density of one-half that of the rest of the network.

Here we report, as in the first test, the resulting average link length and number of nodes in the planar spanning subgraph.

In figure 5.10, we see the average length of link kept in both the Gabriel and Morelia graph. The average Morelia link length has now been improved to be 5.57

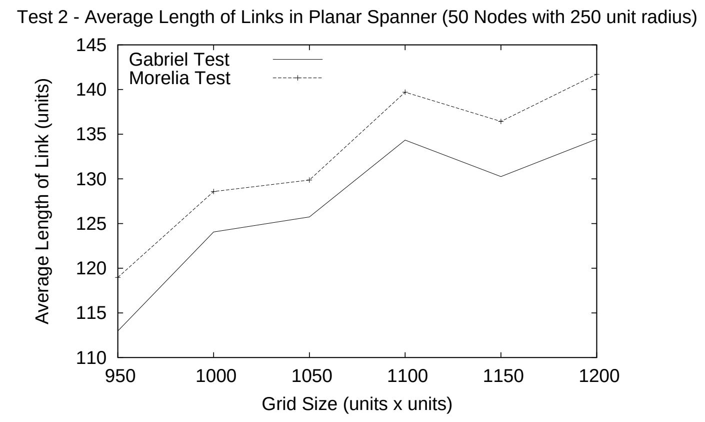

Figure 5.10: Test 2 - Average link length in planar spanning subgraph with Gabriel and Morelia Tests.

units longer than the average Gabriel link.

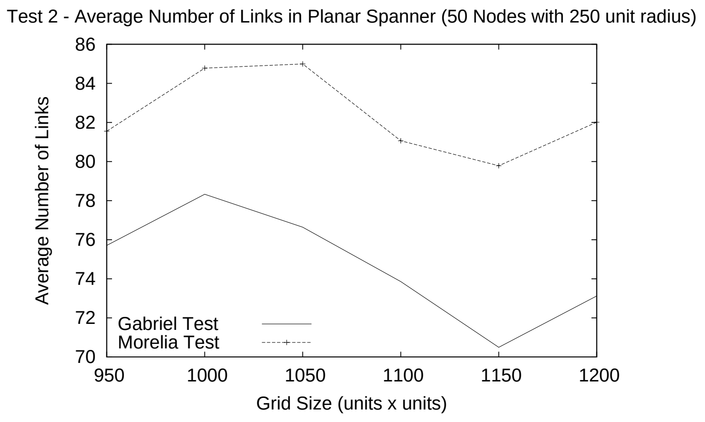

Figure 5.11: Test 2 - Average number of links in planar spanning subgraph with Gabriel and Morelia Tests.

Finally, we show in figure 5.11 the average number of extra links kept in the two graphs. The number has increased to keep 7.67 or 10.33% extra links.

5.6.4 Face Routing Performance Study

In this section we present the simulation results of the face routing algorithm *Face-2* on the same series of random graphs on which the Morelia and Gabriel tests have produced a planar spanning subgraph.

We want to show that, in a network with some sparse areas, the face routing algorithm will perform routing with fewer number of hops on average. Three scenarios were simulated. The first was on a uniform random graph, the remaining two scenarios were on graphs which had different sized sparse regions. In these simulations, the only metrics studied was the *average number of hops* required for face routing.

Scenario 1 - Uniform Random Graph

In this scenario, Test 3, a uniform random network of 50 nodes was built. Each node had a transmission range of 250 units. As before, the grid size was the only parameter changed to vary the network density.

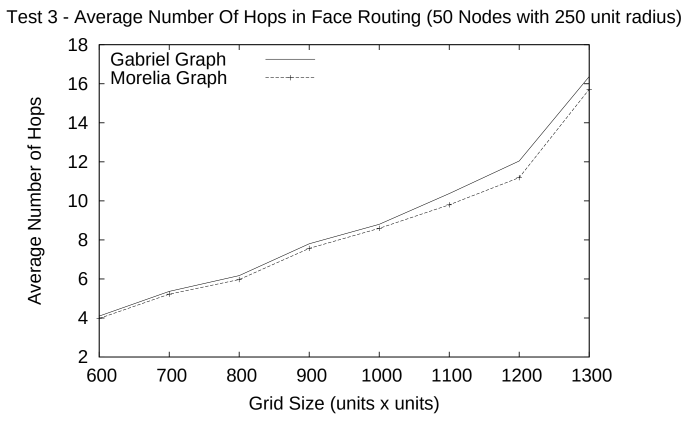

Figure 5.12: Test 3 - Average Number of Hops in Face Routing.

We see in figure 5.12, that in a uniformly distributed network, only minor gains are made in terms of decreasing the number of hops. The average number of hops saved was 0.28 per route.

Scenario 2 - Random Graph with a Sparse Region I

In this scenario, Test 4, we define a region 300x300 units square and set the number of nodes with the region to one-half that of the first test. Routes were chosen that would traverse the region. All other parameters remained as in the previous scenario.

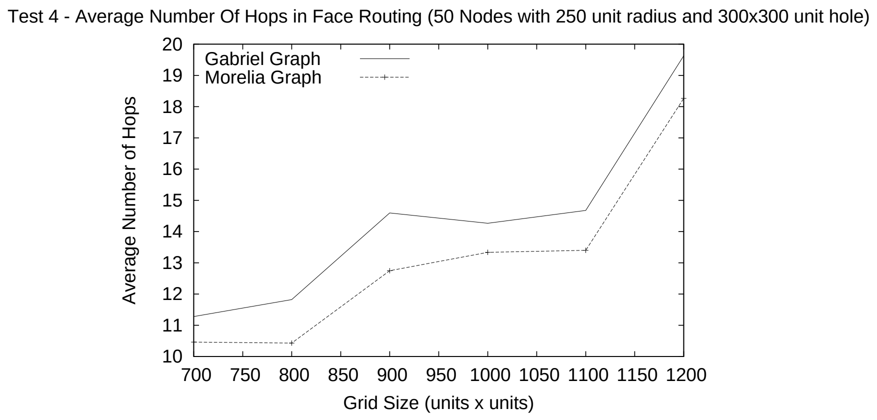

Figure 5.13: Test 4 - Average Number of Hops in Face Routing.

Figure 5.13 shows the results of this simulation. We can see that the face routing algorithms performs modestly better on the Morelia graph. The average number of hops saved in this scenario is increased to 1.44 hops per route.

Scenario 3 - Random Graph with a Sparse Region II

In this final simulation, Test 5, the setup was as in the previous scenario, with the exception of the *hole*. In this scenario, the sparse region was defined to be 500x500 units in size.

In figure 5.14 we can see that there is further improvement of the face routing algorithm. The average number of hops saved with face routing on the Morelia graph has increased to 2.03 hops per route or 10.6%.

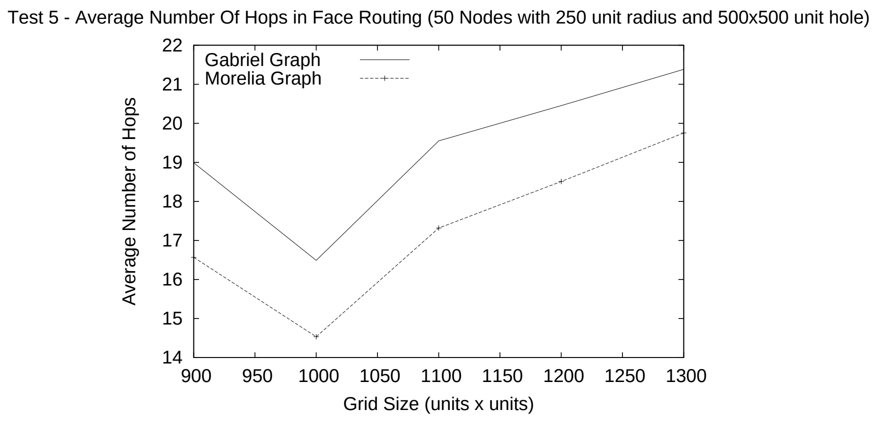

Figure 5.14: Test 5 - Average Number of Hops in Face Routing.

Chapter 6

Face Routing Enhancements

6.1 Introduction

In this chapter, we will look at various modifications to the *face routing* algorithm [6, 23] defined in Chapter 3. First, we will show two approaches to avoiding a region or hotspot defined in the network. Then, we will describe a modification to use face routing for *route discovery* in an attempt to remove loops and the multi-hop effect. Finally, we will define two enhancements to enable devices, which can choose from a fixed number of transmission powers, one of which preserves power consumption.

6.2 Avoiding Dangerous Regions

In this section we will define two similar enhancements to the face routing protocol which will enable the routing of messages from source to target nodes in a network that has a known *dangerous region.* Consider the following: there is a wireless ad-hoc network in operation consisting of many participants. For some reason, you may not wish to pass traffic over nodes owned by certain participants. There could be many reasons to make such a decision not to use these sets of nodes in the region including the following.

1. **Security**. The nodes in question may not be trustworthy. It has been found that nodes in the region are intercepting packets in a malicious manner.

2. **Cost**. By nature of the network, all the nodes in the network are providing a service. There may be a charge for packets sent through other participants equipment. It would be beneficial to avoid those areas.

3. **Unreliable Devices**. It may become aware that there are nodes that are unreliable in a certain region. These nodes may drop packets, enter and leave the network repeatedly and unexpectedly, or may have a failing battery and would not be suitable for routing packets.

Here we define two simple algorithms that can be used to avoid entering a certain region. A region is defined by an ordered list of points $\{(x_1, y_1), (x_2, y_2), .., (x_k, y_k)\}$ where the edges of the region can be found by traversing the list.

6.2.1 Direct Approach

The first algorithm is the direct method. Here face routing operates as normal with the exception that as a node is about to forward a packet, it will check to see if doing so will violate the region defined. In this case, the algorithms will switch to a recovery mode and walk around the perimeter of the region.

Once the packet has been forwarded beyond the forbidden region, face routing can resume from this node. Figure 6.1 shows this algorithm in operation.

6.2.2 Perimeter Approach

The second algorithm is a slight modification of the *direct* approach. Figure 6.2 shows its operation. Here the source node will check if the line from source to target which

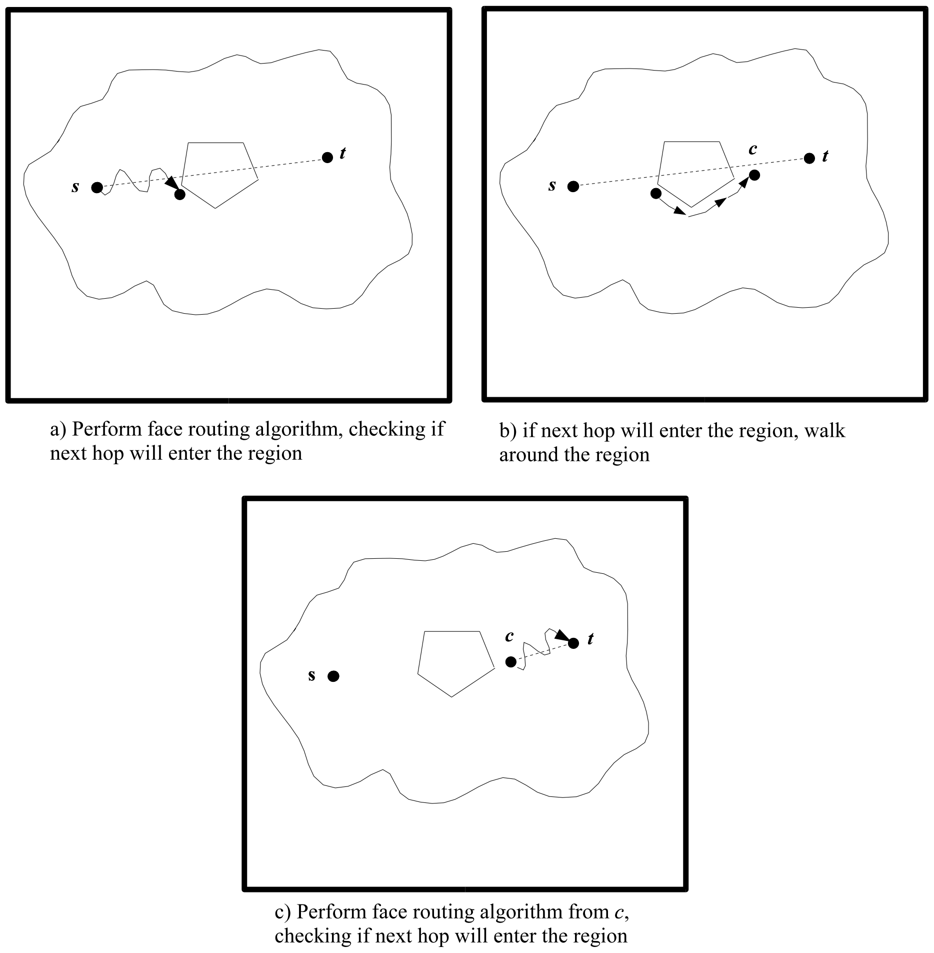
s
t
a) Perform face routing algorithm, checking if next hop will enter the region
s
c
t
b) if next hop will enter the region, walk around the region
s
c
t
c) Perform face routing algorithm from *c*, checking if next hop will enter the region

Figure 6.1: Region Avoidance - Direct Approach.

is used by the face routing algorithm crosses the dangerous region. If it does, it will immediately enter a recovery mode and forward the message in a walk to the perimeter of the network. Once the perimeter is reached, the message is forwarded around the external face of the network until it is determined that the region has been avoided. The algorithm now reverts to the direct approach, described above, to finish routing the message to the target.

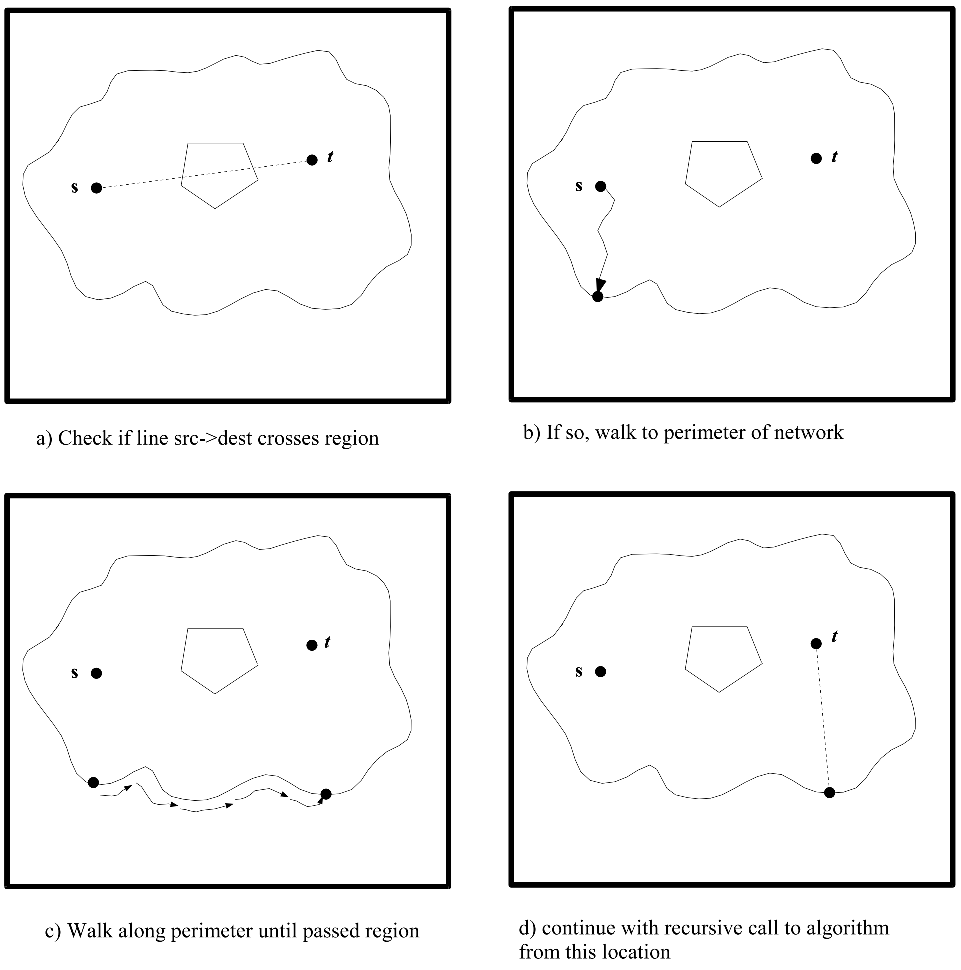
s
t
a) Check if line src->dest crosses region
b) If so, walk to perimeter of network
c) Walk along perimeter until passed region
d) continue with recursive call to algorithm from this location

Figure 6.2: Region Avoidance - Perimeter Walking.

6.2.3 Advantages and Disadvantages

The main advantage of the direct approach is that, as its name implies, it is the most "direct" way to arrive from source to target. Face routing performs as normal, only switching to a recovery mode when it detects it is about to enter the dangerous region. There are two main disadvantages of the direct approach. The first is the fact that at each hop, the algorithm must determine if the next hop will enter the dangerous region. The second is that walking around the perimeter of the dangerous zone is slightly more complicated than with the perimeter algorithm.

The main advantage of the perimeter avoidance approach is that it simply calculates at the start whether the $line(source, target)$ crosses the dangerous region. If so, it will simply forward the messages to the perimeter (external face) of the network. The perimeter of the network is easier to traverse than the dangerous region in the direct approach. There are two main disadvantages of the perimeter approach. The first, is that it will in most cases result in a much larger number of hops in routing messages from source to target. The second is that if the source determines that the $line(source, target)$ does not enter the dangerous region, it must revert to routing using the *direct approach*.

Related Work

The authors in [3] describe an assisted routing technique called *terminode routing*. Part of the algorithm has nodes forwarding packets towards a series of *anchor* points. An intermediate node will forward towards the next anchor node, unless that node is within its transmission range. In that case, the current node will delete it from the anchored path and forward towards the next anchor node.

6.3 Using Face Routing for Route Discovery

This section discusses a slight variation on the location-aware routing model described in Chapter 3. We propose instead that we implement the *face routing* algorithm to *discover* routes from source to target, instead of routing packets. In this scenario, nodes must store, or cache, routes to target nodes with which they wish to communicate. The modified face routing algorithm will be invoked when no route to the target is in the source nodes local cache. The main goals of this algorithm are to:

1. Removes loops, as face routing is not loop-free.

2. Eliminate the multi-hop effect.

In this route discovery algorithm, face routing performs as usual until when traversing a face it intersects the $line(source, target)$. When this occurs, the algorithm will check if the one of the endpoint nodes of the intersecting link is in range of the previous node where we last changed or "flipped" faces. If this is the case, we can simply remove the intermediate nodes traversed around the face, thus eliminating the multi-up effect. The procedure is repeated until reaching the target.

Loops can easily be removed after the complete route has been discovered when the discovery algorithm reaches the target node. The target node will store the optimized route in its cache as well as send it back to the source.

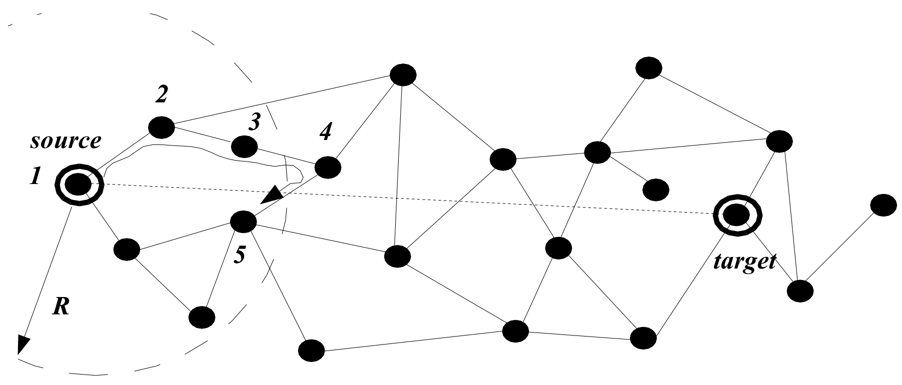

Figure 6.3: Face Route Discovery: Nodes 2, 3 and 4 can be eliminated from the route as node 5 is a neighbour of the source node 1.

Figure 6.3 shows an example of multi-hop elimination in the face route discovery algorithm. Intermediate nodes on the face can be eliminated if a hop resulting in a intersection of $line(source, target)$ is a neighbour of last node where we changed faces.

Related Work

The authors in [9], describe a *shortcut* procedure to enhance the Face routing algorithm. By exchanging two-hop neighbours, nodes can examine their neighbours'

neighbours and calculate the route face routing should take. The current node can avoid forwarding the packet on unnecessary hops if it is in range of neighbours further along the route. This allows for forwarding packets through a region where there are assisting nodes.

6.4 Face Routing with Multiple Transmission Ranges

In this section we will define two interesting algorithms whereby nodes in the network have the ability to broadcast at one of k finite number of transmission power levels. We propose that nodes will determine their neighbours at each of these ranges, and then compute a planar spanning subgraph using any algorithm of choice (*RNG, Gabriel, or Morelia*) at each level.

6.4.1 Goal

Given that devices now have the ability to transmit at k different power levels, we would like to design an algorithm to prolong the life of the network. Densely populated areas of the network could use lower power levels than those in regions that are sparsely populated.

Figure 6.4 shows an example of planar subgraphs of the same underlying network created at two transmission power levels. Note that the first subgraph is not a spanner of the network, but this is not necessarily a disadvantage. If the source and target are both located within the upper area of this network, a route can be discovered using the lower power level. However, if the source and target are in different disconnected regions at at power level 1, the algorithm can use power level 2 to discover a route to the target.

Once these subgraphs have been created one of the following two algorithms may be executed to route packets from source to destination.

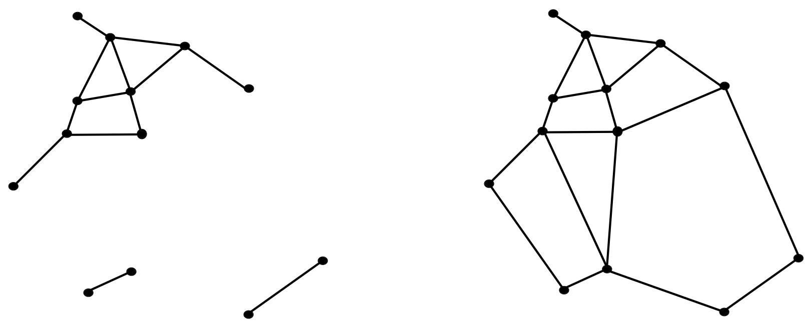

Figure 6.4: Different planar subgraphs created at power levels 1 and 2.

6.4.2 Brute Force

The following describes the naive brute force algorithm called *Multiple Radii Face* routing (*MRF-1*) where a node wishing to communicate with a destination will try to find a route using incremental power levels.

Algorithm: MRF-1

1. Nodes periodically broadcast their information (ID, GPS coordinates) at the highest power level k (*beacon*).

2. All Nodes keep track of their *1*-level to k-level neighbourhoods by listening to these broadcasts

3. Build the planar subgraph for the k different levels.

4. When a source device has a message for some destination device:

   (a) The Face Routing algorithm from source to destination is used using the *level 1* planar subgraph.

   (b) If source receives no reply, it will repeat the process moving to the next power level (levels *2* through k) until a route is established.

   subgraph)

5. Algorithm Complete.

6.4.3 Adding Maximum Hops Per Level

The *MRF-1* algorithm can be further refined. We can introduce a *maximum hop* parameter to the algorithm. Now, we can have more control over the operation of the protocol. Instead of the source node always determining the power level used, this decision can be made by any intermediate node. The source node will commence the routing algorithm as before using the lowest power level, but if the message has not reached the target within the specified maximum number of hops (or the message has reached the source again) the intermediate node will then determine to start the process over at the next power level. We now define *MRF-2* as follows:

Algorithm: MRF-2

1. Nodes know the *maximum hops* per level parameter (this can be passed as part of the message).

2. Nodes periodically broadcast their information (ID, GPS coordinates) at the highest power level k (*beacon*).

3. All Nodes keep track of their *1*-level to k-level neighbourhoods by listening to these broadcasts.

4. Build the planar subgraph for the k different levels.

5. When a source device has a message for some destination device:

   (a) The Face Routing algorithm from source to destination is used using the level 1 planar subgraph.

   (b) If the message does not reach the target within the specified maximum number of hops the intermediate node will move to the next higher power level and resume face routing from its location. This is repeated until target is reached.

6. Algorithm Complete.

6.4.4 Simulations and Case Study

In this section, we discuss the simulation results of our MRF-1 and MRF-2 algorithms. A simulation was designed that allowed the creation of a random network. In this simulation, the grid size was fixed to 500x500 units. Each network consisted of 50 nodes, randomly placed on the grid, each with a multiple ($k = 5$) transmission ranges 20, 40, 60, 80 and 100 units. The planar subgraphs were constructed on the neighbourhoods of the nodes at each of the five transmission ranges.

The MFR-1, MFR-2 and Face routing algorithms were run on the networks to send messages on a series of routes. The Face routing algorithm was only executed on the power level k planar spanning subgraph. The maximum hops per level parameter was modified, and the average power used per route and average number of hops per route were calculated. A simple power model was used. It was based on that the power used is proportional to the distance (transmitted) squared. A receiver must receive a minimal power level c in order to operate, so the power required is $max\{c, d^2\}$. To ensure the minimal power level is received we set $c = power\ at\ level$ 1. Also, there were no maximum number of hops set for power level k as there were no higher power levels available to advance, and the target could be reached at level k.

In figures 6.5 and 6.6 we show the results. The MRF-1 and Face routing algorithms performance is independent of the number hops per level. The MFR-2 algorithm performs significantly better that the MFR-1 algorithm due to the fact that it does not have to restart at the source node at each iteration. The best case in these simulations of the MFR-2 protocol is when the *max hops per level* is set low.

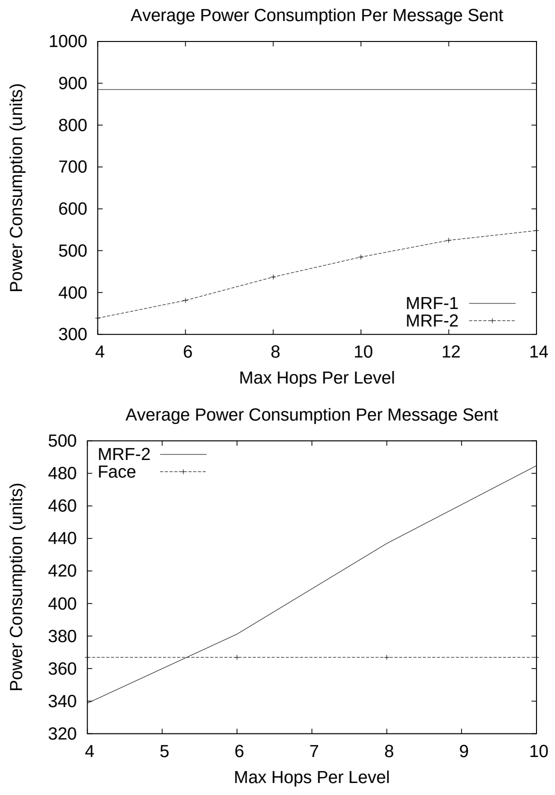

Figure 6.5: Average power used per route with MRF-1, MRF-2 and Face.

The MRF-1 algorithm could be improved if the power level of the last time a packet was sent to a target was stored in a cache for a short period of time. Then, when forwarding the next packet destined for the target, a node could look up the power level in its cache. The first packet sent to target would take a hit on power consumption, but subsequent packets would perform much better, as well as or better than normal face routing.

In our simulation, when the max hops per level is low, MFR-2 uses less power than Face routing. This was due to the fact that in our simulation there were only a few number of neighbours per node at the lower transmission ranges, and the MRF-2 algorithm benefited from moving to the next power level sooner, rather than later. In

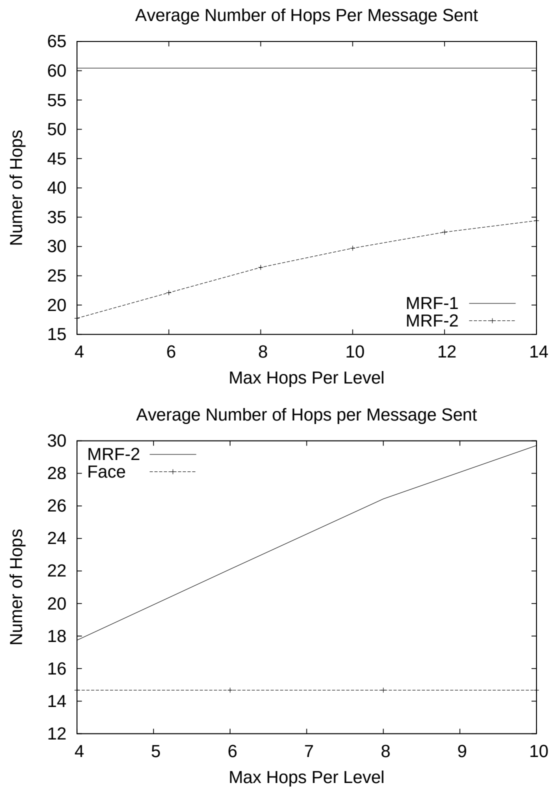

Figure 6.6: Average number of hops per route with MRF-1, MRF-2 and Face.

terms of number of hops per route, Face routing performed with fewer hops per route on average as it was performed at power level k which has the largest transmission range. The MRF-2 uses fewer hops on average than the MRF-1 algorithm as it moves to the next power level (when maximum number of hops at level is met) at an intermediate node, and does not restart at the source node.

Chapter 7

Conclusions and Future Work

In this thesis we have studied routing in wireless ad-hoc networks. Our focus was on location-aware routing algorithms, and work on producing a planar spanner of an ad-hoc network to be used for routing. We also presented some new ideas to enhance the face routing algorithm. We conclude with a summary of our contributions, and discuss some future work.

7.1 Contributions

Chapter 1 In this chapter, we introduced the notion of location-aware routing algorithms and describe the motivation behind our work.

Chapter 2 In this chapter we presented a brief survey of traditional routing algorithms in ad-hoc networks. Many of these algorithms have high overhead both in terms of storage, as well as communications required to maintain a larger picture of the network.

Chapter 3 In this chapter, we began by defining a general model for location-aware routing algorithms. From the model, we see the advantages of location-aware protocols which include reduced overhead for local storage, and reduce communication overhead for maintaining routing information.

Chapter 4 In this chapter, we described some existing algorithms used to create a planar spanning subgraph of an underlying network.

Chapter 5 In this chapter, we presented a new localized algorithm, the *Morelia Test*, to derive a planar spanning subgraph of an underlying wireless ad-hoc network. Through a series of simulations, we have shown that 1) the Morelia test keeps longer links on average than the Gabriel test and 2) the Face Routing algorithm performs better on average in the Morelia graph. We have improved, over the Gabriel graph, the planar spanning subgraph created to be used in face routing.

Chapter 6 In this chapter, we provided some enhancements to the face routing algorithm. We first showed two approaches to avoiding regions of the network, by walking around the defined region. Next, we modified the face routing algorithm to show how it can be used as a routed discovery mechanism to eliminate loops and reduce the multi-hop effect. Finally, we described two algorithms, MRF-1 and MRF-2, which attempt to conserve energy during face routing in a network where nodes can choose from a finite set of transmission ranges. We have shown, through simulation, that the MRF-2 algorithm performs better, as it benefits from moving to the next power level at a node which is closer to the target.

7.2 Future Work

In this section we describe some problems which should be implemented as continuing work in the areas we have studied.

7.2.1 Morelia Test in a Dense Network

As the Morelia test is performed on every link in the network, it may introduce O(n^2) worst case communication messages (in a complete graph) while preprocessing the network. The next step would be to implement and simulate the *leader election* algorithm discussed in Section 5.5 with different values for r, the square size. This should reduce the message overhead of the Morelia test and allow for more efficient operation on a network which is very dense.

7.2.2 Simulation Environment

The simulations were designed and implemented using C++. Although great care was taken in designing and implementing a series of network simulations that could easily be modified to test different scenarios, it became increasingly difficult to make changes to the network as new ideas surfaced. The next step would be to port the simulations to the ns-2 network simulation environment [10].

Also, further experimentation including routing algorithms such as *Greedy-Face-Greedy* or *GPSR*, would show the impact of the Morelia test on these algorithms which depend on face traversals for a recovery mechanism.

7.2.3 Mobility

The next area which should be investigated is to introduce mobility to the simulation network model. All algorithms in the simulations operated on a "snapshot" of the network captured at a single moment in time. By introducing mobility to the situation, the problem of maintaining the Morelia graph becomes an issue. When the nodes are in motion, links may form and break continuously and in a random fashion. An mechanism is needed to decide when the Morelia graph needs to be updated. By moving the simulation environment to the ns-2 environment, mobility models exist to aid in testing any algorithms designed.

7.2.4 Multiple Radii Face Routing

Finally, we would like to investigate the operation of the MRF-2 algorithm in a variety of different network topologies to see if we can achieve better performance. Simulation on larger networks, and networks which have a topology that has dense and sparse regions may show interesting results.

7.3 Final Remarks

We hope that the audience has enjoyed reading these results as much as we have presenting them. Only the future will tell when and where location-aware services will creep into our lives. Someday, when you are using your mobile phone to find the location of the nearest Chinese restaurant in an unfamiliar city, or your mobile phone vibrates (when you enter a mall) to notify you that there is a sale at the electronics store just a few steps away, you will remember this thesis.

Appendix A

Glossary

AODV Ad-hoc On Demand Distance Vector routing.

AFR Adaptive Face Routing.

Beacon A broadcast message sent periodically by all nodes to inform their neighbours of their continued presence.

BFR Bounded Face Routing.

Broadcast A message transmitted by a node that can be heard by all neigbouring nodes.

Connected graph (or network) A graph where for all pair of vertices u and v, a route can be found from u to v.

DSDV Destination Sequenced Distance Vector routing.

DSR Dynamic Source Routing.

Face Closed polygonal region enclosed by the graphs edges.

Flooding A global broadcast, starting from a source node broadcasting to its neighbours, which will repeat the broadcast to their neighbours, which in turn repeat the broadcast to their neighbours, and so on.

Geocasting Sending a message to a set of nodes within a defined region.

GPS Global Positioning System. Devices with a special receiver can determine their geographic location by triangulation with a series of orbiting satellites.

GPSR Greedy Perimeter State Routing.

Hotspot In region avoidance, a hotspot is a region which we do not wish to enter when routing packets.

Killer-app The application which will result in the mass adoption of an emerging technology.

LAN Local Area Network.

Location-aware Any algorithm which makes use of the geographic location of nodes in a network to perform some operation such as routing.

Location Service In location-aware algorithms, nodes must register with such a service so their location information can be found. Nodes wishing to send messages to a target must use a location service to find the targets location.

Multi-hop effect Some algorithms may require sending messages through intermediate nodes even though the message could be sent to a node further on the path directly.

MFR Most Forward within Radius.

Neighbourhood The set of all nodes which are in communication range of a node.

N(v) The neighbourhood of vertex v.

NFP Nearest with Forward Progress.

Packet The smallest unit of data (message) that can be sent from one node to another.

Planar graph A graph where no two edges cross each other.

Planar spanning subgraph A planar graph which includes all nodes in the network.

Progress In Greedy routing, progress refers to whether a neighbour is closer to or further away from the target.

RNG Relative Neighbourhood Graph.

WLAN Wireless Local Area Network.

WLNP Wireless Local Number Portability.

ZRP Zone Routing Protocol.

Bibliography

[1] Lali Barriére, Pierre Fraigniaud, and Lata Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 19–27. ACM Press, 2001.

[2] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84. ACM Press, 1998.

[3] Lj. Blazevic, S. Giordano, and J.Y. Le Boudec. Self organized terminode routing, TR DSC/2000/020, Swiss Federal Istitute of Technology, Lausanne, December 2000.

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

[5] P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opartny, G. Salazar, and J. Urrutia. Morelia Test: Improving the Efficiency of the Gabriel Test and Face Routing in Ad-hoc Networks. *to appear*, 2003.

[6] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *ACM/Kluwer Wireless Networks*, 7(6):609–616, 2001.

[7] G. Calinescu. Computing 2-hop neighborhoods in ad hoc wireless networks. In *Ad-Hoc, Mobile, and Wireless Networks, Second International Conference, ADHOC-NOW 2003, LNCS 2865*. Springer, 2003.

[8] T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi. Performance evaluation of two location based routing protocols. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1678–1687, 2002.

[9] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. In *Proc. IEEE Int. Conf. on Distributed Computing and Systems Workshops, Cluster Computing*, page 461ŋ466, 2001.

[10] Kevin Fall and Kannan Varadhan. The ns Manual (formerly ns notes and documentation). *http://www.isi.edu/nsnam/ns/doc/ns_ doc.pdf, VINT Project*, 2003.

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

[12] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Gemetric Spanner for Routing in Mobile Networks. In *Proc. 2nd Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, pages 45–55, 2001.

[13] S. Giordano, I. Stojmenovic, and L. Blazevie. Position Based Routing Algorithms for Ad hoc Networks: A Taxonomy, July 2001. http://www.site.uottawa.ca/-ivan/routing-survey. pdf.

[14] Z.J. Haas, M.R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for ad hoc networks IETF Internet Draft, draft-ietf-manet-zone-zrp-04.txt, July 2002.

[15] 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.

[16] IEEE. Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-1997*, 1997.

[17] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[18] J. Jubin and J. D. Tornow. The DARPA Packet Radio Network Protocol. *Proceedings of the IEEE*, 75(1):21–32, 1987.

[19] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 243–254. ACM Press, 2000.

[20] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Mobile Computing and Networking*, pages 43–254, 2000.

[21] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 66–75. ACM Press, 1998.

[22] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[23] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[24] F. Kuhn, R. Wattenhofer, and A. Zollinger. Geometric ad-hoc routing for unit disk graphs and general cost models. In *Technical Report 373, ETH Zurich, Department of Computer Science*, 2002.

[25] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric Ad-hoc Routing: Of Theory and Practice. In *Proc. 22nd ACM Int. Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[26] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 24–33. ACM Press, 2002.

[27] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In *Proc. 4^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[28] Tony Larson and Nicklas Hedman. Routing Protocols in Wireless Ad-hoc Networks - A Simulation Study. Master's thesis, Luleȧ University of Technology, Stockholm, Stockholm, 1998.

[29] Xiang-Yang Li, G. Calinescu, P.-J. Wan, and Y. Wang. Localized Delaunay Triangulation with applications in ad hoc wireless networks. *IEEE Transactions on Parallell and Distributed Systems, to appear*, 2003.

[30] Xiang-Yang Li, Ivan Stojmenovic, and Yu Wang. Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Multihop Scatternet Formation. *Accepted for publication. IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2003.

[31] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network Magazine*, 15(6):30–39, 2001.

[32] Patrick Ryan Morin. *Online Routing in Geometric Graphs.* PhD thesis, School of Computer Science, Carleton University, Ottawa, 2001.

[33] J.C. Navas and T. Imielinski. Geographic addressing and routing. In *Proc. 3rd ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '97)*, pages 66–76, 1997.

[34] R. Nelson and L. Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. *IEEE Transactions on Communications*, 32(6):684–694, 1984.

[35] Charles Perkins. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF, Internet Draft, draft-ietf-manet-aodv-00.txt, November, 1997.

[36] Charles Perkins. *Ad hoc Networking (Ed.).* Addison-Wesley, 2001.

[37] Charles Perkins and Pravin 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, 1994.

[38] E.W. Pfeiffer. WhereWare. *Technology Review (An MIT Enterprise)*, http://www.technologyreview.com/ articles/ mag_toc_sept03.asp September, 2003.

[39] Ivan Stojmenovic. Position based routing in ad hoc networks. *IEEE Commmunications Magazine*, 40(7):128–134, 2002.

[40] Ivan Stojmenovic and Xu 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.

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

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

[43] Yu Wang and Xiang-Yang Li. Localized construction of bounded degree planar spanner for wireless ad hoc networks. *Accepted for publication, ACM DialM*, 2003.

[44] Federal Communications Commission Wireless Telecommunications Bureau. Wireless local number portability, http://wireless.fcc.gov/wlnp/, 2003.

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