A GENERALIZATION OF THE FACE ROUTING ALGORITHM TO SOME NON-PLANAR NETWORKS

SABEEL ANSARI

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

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

APRIL 2005

Your file Votre référence
ISBN: 0-494-04439-X
Our file Notre référence
ISBN: 0-494-04439-X

Abstract

A Generalization of the Face Routing Algorithm to Some Non-Planar Networks

Sabeel Ansari

An *ad hoc network* is composed of autonomous and possibly heterogeneous hosts or terminals that communicate with each other over a radio link. Since the radio range of these terminals is limited, the terminals form a *multi-hop* wireless network. These types of networks can work without any supporting infrastructure and are thus self-organizing and adaptive. Each terminal acts both as an end-system and as a router. In this work, we address the issue of unicast routing in such networks.

Since ad hoc networks run on limited resources, traditional routing algorithms designed for wired networks are not suitable for them. Routing algorithms which use the geographical location information of nodes are called position-based routing algorithms. In such algorithms, the ad hoc network is modeled as a geometric graph known as the unit disk graph. The FACE ROUTING algorithm is one such position-based routing algorithm which guarantees delivery without flooding control packets throughout the network. A requirement of FACE ROUTING is that the graph needs to be planar. FACE ROUTING can be used on unit disk graphs by extracting a planar subgraph of the unit disk graph and using only the edges of the planar subgraph for routing. However, the planarization algorithms may fail in some situations, such as in the presence of obstacles, which in turn may cause a routing failure. In this work, we propose an extension to the FACE ROUTING algorithm for a particular set of non-planar graphs, called *Face Routing On Networks with Crossings (FRONC)*. FRONC guarantees delivery on a network that can be obtained from a planar graph by the addition of disjoint crossing edges. It needs $O(l)$ memory, where l is the maximum number of edges in any face in the graph obtained by removing one edge in each pair of crossing edges.

Acknowledgments

At this juncture, I wish to express my gratitude to my supervisors Dr. Lata Narayanan and Dr. Jaroslav Opatrny who have taught me a lot in the past two years. Without their constant backing, support and pressing, I couldn't have completed this thesis. All the discussions that I've had with them have helped me learn how to do research or approach a problem.

I wish to thank my family, to which I dedicate this work, for being such an inspiration and support, especially my brothers Najeeb and Furquan. I would also like to thank my colleagues Anup Patnaik and Israat Tanzeena Haque for all the technical discussions we have had. I would also like to thank Dr. Kranakis and Paul Boone of Carleton University for sharing their information with me.

And last, but definitely not the least, I would like to thank each of my close friends who have made me feel at home here in Montreal: Gurudath - for spending 18 hours a day with me and sharing everything, Anup - for his eccentricities, Ram - for being confused and being a sport while being laughed at, Tejas - for just being Tejas, Israat - who is fun to be with inspite of all her complaining and Kruthi- for being a good sport.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Wireless Networks

In recent times, the convenience and availability of inexpensive consumer electronics and computers has led to the mass utilization of such devices. One category of these hi-tech devices with widespread usage are the wireless products. Wireless devices use infrared or radio frequency waves for communication. One way of classifying wireless technologies is based on the application of the technology (i.e., depending on where they are used). One such classification is:

1. *Voice and messaging devices* such as cell phones, pagers and other two-way business radio devices.

2. *Handheld and internet-enabled devices* such as internet-enabled cell phones and PDAs (Personal Digital Assistants).

3. *Data networking devices* such as WLAN (Wireless Local Area Network) products.

Our interest in this work focuses mainly on the third category although it has applications in the second category. WLAN is mainly based on the IEEE 802.11 standard (Bluetooth is another competing technology). Originally started as an extension of a wired network, fully-fledged stand-alone wireless networks were later developed. Such a (completely) wireless network can be built in two ways. Firstly, using an *Access Point* and providing some base for the communication to take place and secondly, to let the wireless devices communicate amongst themselves without any support. The latter category of wireless networks are called *ad hoc networks*, which are the focus of this thesis.

In the early 1970s, the U.S. DoD (Department of Defense), soon after the development of packet switching technology, sponsored research to enable packet switching technology to operate without any restrictions of fixed or wired infrastructure. The original motivation was for battlefield survivability, mainly a military objective. Later on, more commercial applications were discovered/invented. The ALOHA project at the University of Hawaii demonstrated the feasibility of using radio terminals to send/receive data packets (but only for a single hop). The ALOHA project led to the development of *multi-hop* packet radio networks (PRNET), sponsored by the Advanced Research Project Agency (ARPA). Routing was achieved by using the classic Bellman-Ford algorithm [FF62] which is basically a distance vector algorithm. A distance vector routing algorithm is a decentralized routing algorithm where routers advertise their information to their neighbors. Routing tables present in each router are updated to find the shortest path to a destination.

PRNET led to the development of Survivable Adaptive Radio Networks (SURAN) in the early 1980s [FL01]. SURAN improved on the radio technology, scalability of algorithms and resilience to some of the electronic attacks. More public interest was attracted in such networks due to the availability of notebook computers and open-source software in the early 1990s. The IEEE 802.11 subcommittee adopted the term "ad hoc networks" soon after some routing protocols for such networks were presented ([PB94], [Joh94]). This resulted in the birth of *ad hoc networks*. The DoD funded programs such as Global Mobile Information Systems (GloMo) and Near-term Digital Radio (NTDR), which contributed more to the field.

1.2 Ad Hoc Networks

An ad hoc network, more commonly known as a *mesh network* in the industry, is composed of autonomous (possibly heterogeneous) hosts or terminals that communicate with each other over a radio link. The absence of infrastructure is the defining characteristic of such networks. It is not practically possible to have a very large transmission radius for each terminal. This is because the spectrum assigned limits the transmission range of the devices in order to avoid radio interference. The fact that the power consumption increases with distance between two terminals also limits the range of transmission of the devices. The terminals thus need to have a moderate range and form a multi-hop network, in which each terminal can forward packets only to a few selected terminals in its radio

range. Each terminal in the network thus acts as an end-system and also, at the same time, as a router, forwarding packets to assist in the existence of the network. Ad hoc networks may use MAC (Media Access Control) protocols such as IEEE 802.11, Bluetooth or other frequency-hopping technology to communicate. An ad-hoc network is *self-organizing* and *adaptive* [Toh02]. This implies that when terminals move, the network topology changes. The communication protocols involved need to adapt to these changes. The term "ad hoc" implies that the network can "take on different forms" and "can be mobile, standalone or networked" [Toh02]. Generally, the bandwidth of these networks is low when compared to wired networks.

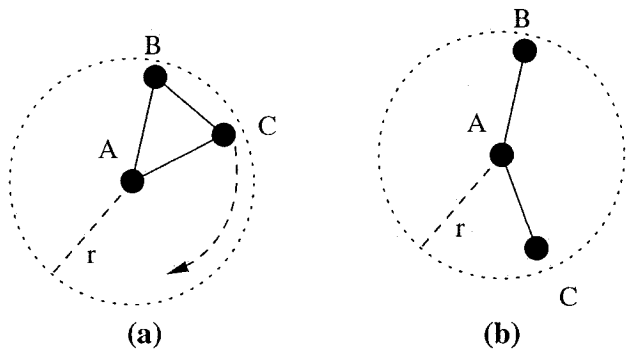


Figure 1.1: Topology of network changes when hosts move in ad hoc networks.

Figure 1.1 illustrates how the topology changes in ad hoc networks when hosts move. A, B and C are mobile hosts, the circle around A depicted by radius r represents the radio range of A. The lines connecting two hosts represent that the two hosts can communicate with each other directly.

A wireless *sensor network* is one form of ad hoc wireless network. Sensors are minute in size and possess both communication and storage capabilities. These can be used in battlefield operations and in other industries such as agriculture, food, emergency-operations which can be bio-hazardous etc. Sensor networks can be used to perform some computation based on the collected data, which can in turn be used to derive some statistic.

Spurred by the interest in ad hoc networks, the IETF set up a working group to work on developing and standardizing ad hoc networking technology. This working group was called MANET. The MANET working group "is a chartered working group within the Internet Engineering Task Force (IETF) to investigate and develop candidate standard Internet

routing support for mobile, wireless IP autonomous segments".

The NCCR's MICS is a collaboration between universities and the industry launched in 2001 and is based in Switzerland. The word *terminode* is formed by the conjoining the words 'Terminal' + 'Node'. The terminode project has been responsible for an appreciable number of inventions in the field.

1.3 Ad-hoc Networking Issues

Networking by itself is a multilayer problem. Apart from the problems involved in networking, there are additional issues that arise and need to be addressed in wireless networks and in particular ad hoc networks. The physical layer must adapt to changes in link characteristics such as collision of signals, radio propagation problems etc. The MAC layer needs to minimize collisions, allow fair access, contend with the hidden and exposed terminal problems. The network layer needs to gather and distribute information for the packets to be routed to the prescribed destination. This has to be done with efficiency as the topology can change often and also since the bandwidth is scarce. The protocols also have to face the problem of integrating the network with either a non ad-hoc wireless network or with a wired counterpart, if they exist. The transport layer needs to handle delay and packet loss and hence conventional algorithms designed for wired networks cannot be used without any modifications. At the application layer, applications need to be resilient to frequent disconnections and reconnections with peer applications. Apart from these, there also exist other issues such as security (at more than one level of the stack), energy efficiency, quality of service (QoS) etc. In this thesis, we restrict our attention to the network layer, in particular to unicast routing algorithms for ad hoc networks.

1.4 The Routing Problem

We concentrate on the unicast routing problem in ad hoc networks in this work. The *unicast routing* problem is defined as discovering a route to deliver packets from one source to one destination via other terminals in the network. Other routing problems that have been studied are broadcast, multicast and geocast. Some of the classical routing algorithms in wired networks use routing tables (distance vector or link state routing) to achieve routing. This is impractical in ad hoc networks, as they run on limited bandwidth resources and

have power constraints. Besides, such routing algorithms take a long time to converge and this may not be practical in ad hoc networks as the network topology can change dynamically and frequently. Another important difference in ad hoc networks as opposed to other networks is that there does not exist any fixed infrastructure such as routers. One way to achieve routing in ad hoc networks is to use the geographical location information of the nodes. Our work in this thesis focuses on using this method.

1.5 Contribution of Thesis

This work concentrates on position-based routing algorithms, where each terminal has knowledge of the exact geographical position of itself and its neighbors. For unicast routing algorithms, we assume that a source terminal is also aware of the destination's geographical position. Routing algorithms have been designed for such a scenario and there exist algorithms which guarantee delivery. One such algorithm which guarantees delivery is called the FACE ROUTING algorithm. This algorithm is designed for planar graphs, but can be used on other graphs representing ad hoc networks, provided they are planarized. The main contribution of this thesis is an extension of the FACE ROUTING algorithm for some non-planar graphs. We also study the experimental performance of our algorithm and compare it with that of FACE ROUTING . This work is one of the initial steps towards designing routing algorithms for non-planar graphs.

1.6 Organization of Thesis

In Chapter 2, we start with a discussion on graphs, geometric graphs and their planar subgraphs. We later describe some well-known routing algorithms for ad hoc networks, with an emphasis on position-based routing protocols. The FACE ROUTING algorithm is discussed in detail as this work mainly deals with it.

In Chapter 3, we start with the discussion of our working model, the motivation and a formal definition of a class of non-planar graphs of interest. This is followed by a description of our new algorithm called Face Routing On Networks with Crossings (FRONC) and a proof of its correctness. We end the chapter with a discussion on some of the limitations of FRONC .

In Chapter 4, we give details of the simulation setup with a thorough account of how

the crossing edges are generated in our simulator. The results of the simulation accompany the discussion to end the chapter.

In Chapter 5, we conclude the thesis with a short discussion of future work.

Chapter 2

Preliminaries

In this chapter, we give the background required for our new routing algorithm. Ad hoc networks are generally represented as geometric graphs. Hence, a discussion on graphs, geometric graphs and the planarization algorithms involved is necessary before we discuss routing algorithms in ad hoc networks. Section 2.1 defines combinatorial graphs. In Section 2.2, we define geometric graphs followed by a discussion on planarizing geometric graphs. Section 2.3 gives an introduction to routing in ad hoc networks followed by a classification of routing algorithms in Section 2.4. As this thesis concentrates on the FACE ROUTING algorithm, a detailed discussion on it is presented in Section 2.5.

2.1 Graphs

Networks have traditionally been represented as graphs, and so, graph algorithms such as computation of shortest path between two nodes, have had direct applications in networks.

A graph can be defined as follows (from [Car79]).

A graph $G = (V,E)$ consists of

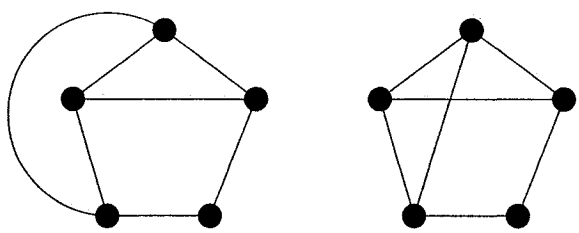
1. a finite set $V = \{v_1, v_2, \ldots, v_n\}$ whose elements are called *nodes* or *vertices*;
2. a subset E of the Cartesian product $V \times V$, the elements of which are called *edges*.

A graph G is said to be *planar* if there exists some geometric representation of G which can be drawn on a plane such that no two of its edges intersect [Deo74]. A graph that

cannot be be drawn on a plane without a crossing between its edges is called *nonplanar* [Deo74]. Note that the "meeting" of edges at a vertex is not considered an intersection.

2.2 Geometric Graphs

The definition of a graph in Section 2.1 addresses a class of graphs called *combinatorial* graphs. Combinatorial graphs can be embedded in the plane by assigning coordinates to nodes. The edges can then be drawn on the plane in any way, and are not required to be straight line segments. See Figure 2.1(a) for a planar embedding of a graph. This is not a realistic representation of an ad hoc network, as the wireless link between two nodes should be modeled by a straight line. So, to represent ad hoc networks, we adopt a class of graphs called geometric graphs. Thus, an ad hoc network is represented by a *geometric undirected graph*, $G = (V, E)$, with vertices representing mobile hosts, and an edge connecting any pair of hosts that can communicate directly. The set of vertices V is thus a set of points in the Euclidean plane. In the geometric representation, an edge is a straight line segment between its end-points. We denote the line segment from point a to point b by $\overline{ab}$. All references to a graph in this work, unless otherwise stated, refer to a geometric undirected graph.



(a) Combinatorial **(b) Geometric**

Figure 2.1: Graph representations.

Figure 2.1(b) shows that with the same node positions as in Figure 2.1(a), the geometric graph is non-planar. Thus, a graph may be planar when represented as a combinatorial graph, but is not planar when we consider the corresponding geometric graph.

2.2.1 Unit Disk Graph

When all nodes in an ad hoc network have identical transmission ranges, it can be modeled by a specific type of geometric graph called a *unit disk graph* (UDG). The vertices and the edges in the graph model a scenario where vertices are the mobile hosts and the edges represent their wireless transmission range. As shown in Figure 2.2(a), an edge exists (i.e., neighboring hosts can communicate) in the graph when the Euclidean distance between two vertices is less than or equal to a host's transmission range, denoted by r. Figure 2.2(b) shows the UDG for the hosts in Figure 2.2(a). The following is the formal definition of a unit disk graph.

> An edge exists between two nodes u and v if and only if the Euclidean distance between u and v is at most 1.

A unit disk graph corresponds to a situation in which all nodes are identical devices. There does exist the possibility of devices having irregular transmission ranges (i.e., the transmission range of a host is not a disk) possibly due to obstacles. Also, an ad hoc network can be composed of different devices which could have different transmission ranges which can depend on the range of the antenna, battery power, etc. In [BFNO03], a network model containing nodes with *irregular transmission ranges* is considered. Every node is connected to all nodes within distance r, not connected to nodes at distance greater than R and may or may not be connected to nodes between distance r and R. An algorithm is presented which guarantees delivery when the ratio of the largest transmission range (R) to the smallest transmission range (r) is at most $\sqrt{2}$.

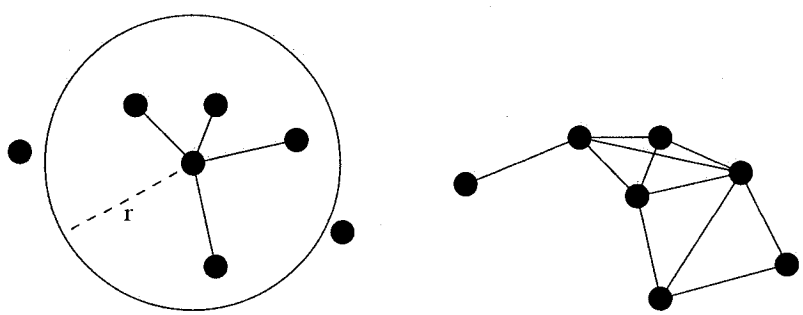


Figure 2.2: The Unit Disk Graph.

A similar model called a *quasi unit disk graph* is presented in [KWZ03]. The model contains all edges shorter than a parameter d between 0 and 1 and no edges longer than 1. A message complexity lower bound for a volatile memory routing algorithm (a routing algorithm where each node on the route holds $O(logn)$ bits) is presented and it is shown that a flooding algorithm matches the lower bound and is asymptotically optimal with respect to the message complexity.

2.2.2 Planarization of a Graph

As FACE ROUTING guarantees delivery when the underlying graph is planar, a look at planarizing graphs is necessary at this point. Planarizing involves removal of certain edges from a unit disk graph such that the resulting graph is planar. As ad hoc networks have limited resources and there is a need to keep the overhead communication to a minimum, it becomes necessary for planarizing algorithms to be distributed in nature. A detailed discussion of two common planarizing algorithms follows.

Gabriel Graph

One of the most widely used planarizing algorithms is the *Gabriel graph* (GG) construction. A distributed algorithm to find the Gabriel subgraph of a graph is easy to implement as each host requires information only about its immediate neighbors. Each host executes this algorithm and removes some edges from a unit disk graph. It has been proved that the Gabriel subgraph of a unit disk graph is connected and planar , provided the unit disk graph itself is connected (*Lemma 1* from [BMSU99]). Gabriel graphs can be defined as follows [GS69]:

> An edge (u,v) exists between vertices u and v if no other vertex w is present within the circle whose diameter is $d(u,v)$, where $d(u,v)$ represents the distance between nodes u and v. In equational form,
>
> $$\forall w \neq u,v : d^2(u,v) < [d^2(u,w) + d^2(v,w)] \quad (1)$$

In Figure 2.3 (a), the edge (u,v) will exist because x is outside the circle with diameter $d(u,v)$.

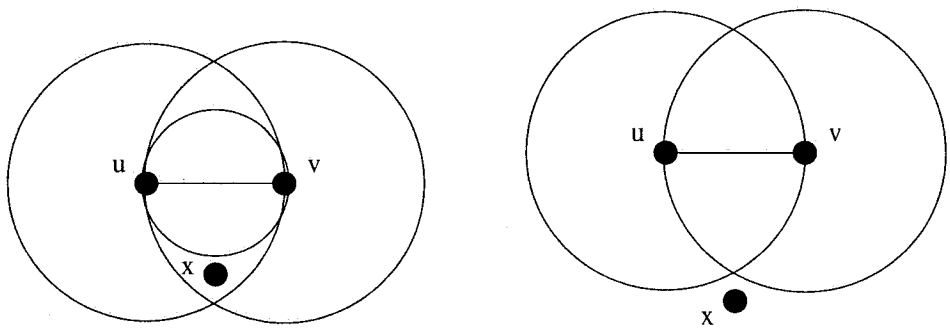


Figure 2.3: The Gabriel and Relative Neighborhood graphs.

Relative Neighborhood Graph

Another commonly used planarization algorithm is the *relative neighborhood graph* (RNG) construction presented in [Tou80]. As with the Gabriel graph, the RNG construction is also simple and distributed in nature. The RNG can be defined as follows [Tou80]:

> An edge (u,v) exists between vertices u and v if the distance between them, $d(u,v)$, is less than or equal to the distance between every other vertex w, and whichever of u and v is farther from w. In equational form,
>
> $$\forall w \neq u,v : d(u,v) \leq max[d(u,w),d(v,w)] \quad (2)$$

In Figure 2.3 (b), edge (u,v) will exist because x is outside the lune formed by the intersection of the circles.

It is important to keep in mind that these planarization algorithms are based on the unit disk graph, which is based on the assumption that all devices in the network have a uniform transmission range. The planarization algorithms (RNG and GG) differ in only one aspect as to what distance they use to remove edges from UDG. Since the area of coverage (for removal of edges from UDG) is comparatively smaller in GG, it removes fewer edges. Thus Gabriel graphs are more dense than RNG. The length of path computed using a routing algorithm on GG is generally shorter as compared to RNG. This is the reason we use Gabriel graphs in this work.

2.3 Routing in Ad Hoc Networks

Unicast routing is an important issue in the design of any internetwork. It involves the forwarding of packets from one terminal or subnet to another such that the packets are delivered to the desired destination. A routing algorithm is expected to provide a definitive result of either discovering the path to the destination or failure. The problem of routing in ad hoc networks is of great interest among researchers. This is because the routing problem is more challenging in ad hoc networks since the terminals are autonomous and there seldom exists a central authority to control anything. Numerous routing algorithms have been proposed. Some of the desired characteristics have been identified and pointed out in [MC98]. They are loop-freedom, distributed operation, path strategy, metrics, memorization, guaranteed message delivery, scalability and robustness. This chapter takes a look at routing algorithms, their classifications with an emphasis on position-based routing. This chapter also discusses the graphs involved in this work.

2.4 Classification of Routing Algorithms

Routing protocols have been classified into *topology based routing* and *position based routing* [MWH01]. Topology based routing is further divided into proactive, reactive and hybrid approaches depending on when and how the routing mechanism is invoked.

2.4.1 Topology Based Routing

Proactive routing algorithms employ the traditional approach such as distance vector or link state routing, where the routes from each host to every other host are pre-computed, regardless of the routes' requirements. Some of the proactive routing algorithms are **DSDV** (Destination Sequenced Distance Vector routing) [PB94], **TBRPF** (Topology Dissemination Based on Reverse Path Forwarding) [OTL04] etc. In these algorithms, topology information is exchanged on a regular basis to update the routing tables. A routing table is present in each host which needs to converge for proper operation. Routing tables contain a (destination, next hop neighbor) pair for each possible destination host in the network. Because this category of protocols require a lot of message exchanging and takes a long time to converge, they are impractical and inefficient in ad hoc networks.

Reactive routing algorithms, as the name suggests, discover the route to the destination only when required. When a host needs to communicate with a particular destination, it invokes the route discovery phase. The method of path discovery and the information convergence at the hosts depends on the specifics of an algorithm. A route maintenance phase also exists in majority of the reactive routing algorithms which tries to keep track of host movements in order to keep the information updated. **AODV** (Ad hoc On-Demand Distance Vector) [PR99] and **DSR** (Dynamic Source Routing) [JM96] are two frequently cited reactive routing algorithms.

DSR is a source routing algorithm. The source constructs a header for the packets which contains the route to be taken to reach the destination. Each transit host checks the header and forwards the packet to the corresponding neighbor. In order to discover a route, a source broadcasts a *route request* (RREQ) packet which is re-broadcasted by other hosts. When a route is successfully discovered, a *route reply* (RREP) packet containing the route is sent back to the source. Route maintenance is provided by using *route error* (RERR) packets. The upstream host of link-break propagates a RERR packet to the source, which results in the truncation of that route in the source's route-cache. A new route discovery ensues.

AODV also uses *route request* packets during its route discovery phase. RREQ packets are rebroadcast by hosts which do not have a route to the destination. While this is done, backward pointers are set up. Once the destination has been discovered, a *route reply* packet is sent back to the source. As the RREP propagates back to the source, hosts set up forward pointers to the destination. When a link fails, a *route error* packet is sent to the source by the upstream host on the link-break. DSR has a potentially larger control overhead and memory requirement than AODV since packets in DSR have to carry the full route information when a route is newly discovered; whereas in AODV, only the next-hop information is required as the route is set up using the backward pointers.

There exists a category of **hybrid routing algorithms**, in which the hosts are assigned to zones. Routes are pre-computed (pro-active) for all the hosts inside a zone. Thus a proactive approach is taken for all intra-zonal communication and a reactive approach for all inter-zonal communication. **ZRP** (Zone Routing Protocol) [HP99] and **HARP** (Hybrid Ad hoc Routing Protocol) [NBN01] are two such hybrid routing protocols.

2.4.2 Position Based Routing

Position based routing has been called *online routing* ([BMSU99] and [BM99]), *location-aware routing* or *geographical routing* ([JPS01]) in the literature. The information required at each host is only the current host's coordinates, the neighbor's coordinates and the destination's coordinates. This information can be obtained by using a low-cost GPS device or by using some other positioning mechanism such as location-service algorithms. A couple of ways to do this is by using the Grid Location Service (GLS) [LJD+00] or by using an algorithm given in [CHH01] which uses the distances between the hosts to build a relative coordinate system in two dimensions.

Once the sender learns the destination's geographical location, it can include this information in the data packets in the form of a header. Each transit host then forwards the packets based on this location information until the packet reaches the destination. Position based routing, thus, does not need to discover routes before transmitting the packets, and also, does not need to maintain any routes at any point of time.

Advantages of Position-Based Routing

Using geographical location information gives the routing algorithms an advantage over traditional routing methods. The following are some of the advantages.

- *Scalability.* Proactive and reactive routing algorithms can have scalability problems. Since wireless devices work on limited resources, frequent exchange of control messages and the size of routing tables can slacken the network. Also, the size of the routing tables is proportional to the size of the network. However, in position-based routing, it is neither necessary to maintain explicit routes to the destination nor is it necessary to *flood* the network. Flooding is essentially broadcasting where packets are forwarded to all neighbors. Position-based routing, thus scales well [MWH01].

- *Geocasting.* It is easy to implement geocasting: a technique in which packets are delivered to all the hosts in a certain geographical area. The region to deliver packets is specified by a point and radius. All hosts contained in that area are the receivers. For more information, see [NI97] and [Mai04]. In some special cases, this can be seen as broadcasting (where the radius of packet delivery is infinity) or multicasting.

- *Security.* Some of the common attacks which affect topology-based routing algorithms, such as a rogue node advertising a shorter route to the destination and dropping the packets or manipulating information in the packets, cannot be carried out.

Classification of Position-Based Routing Algorithms

The forwarding scheme at each transit node is determined by the particular algorithm. Position-based algorithms can thus be classified into three categories (from [MWH01]). Figure 2.4 illustrates the forwarding scheme for the progress based schemes. Note that the dashed line connecting S and D is not an edge in the ad hoc network, but a reference line. r is the transmission radius of S.

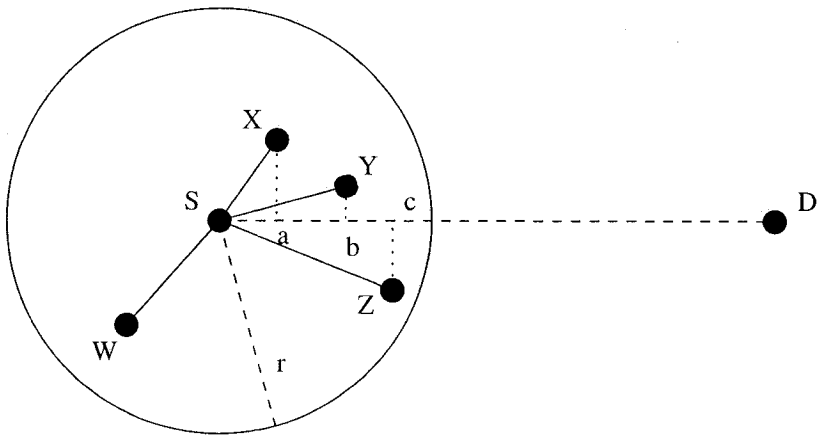


Figure 2.4: Progress-based forwarding schemes.

1. *Progress-based schemes*

   The packets are forwarded in the direction of the destination to reduce the absolute distance, based on some predetermined heuristics. In **MFR** (Most Forward within Radius) [TK84], packets are forwarded to a neighbor closest to the destination in order to achieve maximum progress. In Figure 2.4, S forwards the packet to Z since it achieves the maximum progress, as can be seen by the projection 'c'. This is closely related to **Greedy** routing, where the packet is forwarded to the neighbor whose distance to the destination is smallest among all neighbors. A contrary scheme is sometimes used, called **NFP** (Nearest with Forward Progress) [HL86]. Here, the

packets are forwarded to the closest neighbor, in Figure 2.4, S would forward the packet to X. The idea behind this scheme is to adjust the transmission power to reduce the (probability of) number of packet collisions. In **compass routing** [KSU99], the main goal is not to reduce the distance to the destination, but rather, to reduce the angle. In other words, packets are forwarded to a neighbor who is closest to the line connecting the forwarding node and the destination. In Figure 2.4, S would forward the packet to Y as it is closest to the S-D line.

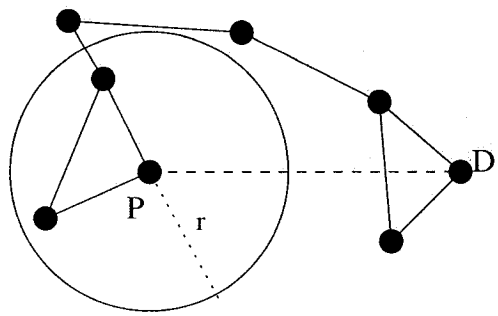


Figure 2.5: Greedy routing fails after reaching P.

The progress-based schemes are known to fail in various scenarios. The greedy algorithm could end up at a local maximum - a host which is closest to the destination relative to all its neighbors. The only path from that host to the destination would be to go through a host which is relatively farther (see Figure 2.5). Compass routing is also known to fail in certain scenarios. As described in [KSU99], compass routing could result in the packets being forwarded in a cycle.

2. *Face routing*

To overcome failures of progress-based algorithms, an algorithm is given in [KSU99] by the name of *compass routing 2*. In the literature it is known as *perimeter routing* but more commonly known as FACE ROUTING . This algorithm guarantees delivery if the underlying graph is planar. An optimization of this FACE ROUTING algorithm (called *face-1*) is given by [BMSU99] and is called *face-2* (the optimization is: face-1 terminates in at most $4|E|$ steps and face-2 reduces that to $3|E|$; where $|E|$ is the number of edges in the graph G). Since most of the work in this thesis is regarding FACE ROUTING , a more detailed discussion follows in Section 2.5.

3. *Combining progress based schemes with face routing*

   Since FACE ROUTING seldom takes the shortest path to the destination, [KK00] and [DSW01] propose using a combination of the greedy approach and the FACE ROUTING algorithm. They propose using the greedy algorithm as the norm and when stuck at a local maximum - use face routing to get out of it. Once the local maximum is taken care of, it can revert back to the greedy procedure. [DSW01] call it the **GFG** (Greedy-Face-Greedy) algorithm. [KK00] propose this idea as a protocol and call it **GPSR** (Greedy Perimeter Stateless Routing).

   In [KWZ02], the authors propose executing the face routing algorithm inside an area bounded by an ellipse. It follows work on the original face routing, so the algorithm goes around the whole face until it finds the best point to cross-over to the next face. If a point cannot be found inside the bounded ellipse, it goes back to the beginning of the face (back to the point where it entered the face) and increases the size of the ellipse to look for the new point of face-transfer. They call it the **AFR** (Adaptive Face Routing). In [KWZ02], the authors show that any geometric ad hoc routing algorithm is quadratic in the cost of an optimal path, which holds true for the Euclidean distance, the link distance and the energy metric. They prove that AFR matches the lower bound shown for the unit disk graph.

   $GOAFR^+$ [KWZZ03] (pronounced "*gopher plus*") combines the greedy and face routing approach. $GOAFR^+$ is an asymptotically optimal algorithm in which the authors propose routing inside a circle constructed with the destination as the center and the radius being equal to $(\rho_0|st|)$ where $\rho_0 = 1.4$, is a constant. With this, $GOAFR^+$ tries to stay inside the circle so as to limit the algorithm inside these bounds. When in the face-routing mode, $GOAFR^+$ uses an "early fallback" mechanism to try and revert back to greedy mode as soon as possible.

4. *Restricted flooding*

   Restricted flooding, also called *partial flooding*, is a mechanism in which packets are forwarded to more than one node which lie in the direction of the destination. The selection of nodes is based on some heuristics particular to an algorithm/protocol.

   - In **DREAM** - Distance Routing Effect Algorithm for Mobility [BCSW98], information about location and speed of the destination are used to select nodes to which packets are to be forwarded. The neighbors chosen are specified by

an angle, which depends on the last known position of the destination and an expected region in which the destination would be present, as it is free to move. The authors describe the *distance effect* according to which, nodes appear to move slower as the distance is increased. In other words, the change in angle is bigger when a node is closer as compared to another node which moves the same distance and at the same speed, but is farther. If no neighbors are present in the specific direction, then a recovery procedure has to be invoked, the actual implementation of which is not discussed in [BCSW98]. The authors propose resorting to partial or complete flooding for the recovery.

- In **LAR** - Location Aided Routing [KV98], a *request zone* is computed by the source. This is usually computed based on information from earlier communication with the particular destination. If no such information exists, LAR is reduced to simple flooding. All nodes that do not belong to the request zone do not forward the route request packets to their neighbors. Typically, the *expected zone* of the destination is present inside the request zone. Two different schemes of request zones have been defined. See [KV98] for full specifications.

5. *Hierarchical routing*

   Hierarchical routing has been seen as a solution for scalability in traditional networks. It therefore seems to be an interesting option to consider in ad hoc networks. Here, long-distance routing is usually performed based on location information and for all local routing, a more pro-active approach is used.

   - **The terminode approach**

     The Terminode approach [BBC+01] uses a combination of proactive routing and greedy/face routing. Two routing protocols are used, viz. **TLR** (Terminode Local Routing) and **TRR** (Terminode Remote Routing). When the distance to the destination is small (called *TLR-reachable*), TLR is used. TLR is essentially a proactive protocol which stores all nodes reachable from the node in question using the TLR protocol. The current implementation is set to two hops[1]. TRR is used in cases where the destination cannot be reached using the TLR (the *non TLR-reachable* nodes). TRR uses either **AGPF** (Anchored Geodesic Packet

[1] A demo exists on the MICS website - http://www.mics.org/

Forwarding) or **GPF** (Geodesic Packet Forwarding) to forward the packets depending on the availability of anchors. Anchors are geographical points (it can also be a node) in the working area.

AGPF: When anchors exist, TRR uses AGPF. The computation of anchors is performed by the source node using the path discovery method called **FAPD** - Friend Assisted Path Discovery (see [BBC+01] for details). The source specifies the anchors that the packet should pass through (or pass by). This, in effect, makes TRR position-based source routing. For the actual forwarding, GPF is used with packets directed towards the next anchor along the path.

GPF: GPF is a simple greedy method which forwards the packets towards a particular geographical location. This geographical location can be an anchor or the destination. The greedy method used is similar to MFR where a neighbor is chosen which achieves maximum progress. In cases where the packets might reach a local maximum due to obstacles or "Terminode desert", routing along the perimeter is used as described in GPSR/GFG.

- **Grid routing**

  The GRID project[2] at MIT uses a technique similar to that of the terminode approach, where all short distance messages are delivered using a proactively built table; and all long range messages are forwarded based on the geographical location, here, in a grid-by-grid manner [DM01]. The geographical area is divided into a number of squares, called *grids* (see [LJD+00]). Each grid has a leader which *has* to be position-aware. The other nodes in that grid may or may not be position aware. This is similar to the concept of dividing nodes into *clusters* - with each cluster having a leader.

- **Clustered routing** The nodes are divided into clusters: as cluster members or elected clusterheads to form a hierarchical routing environment. One clustered routing algorithm differs from another in the specifics of its clusterhead election and the routing decisions taken for routing between clusters. Some clustered routing algorithms are **CBRP** (Cluster Based Routing Protocol) [JLT99], **WCA** (Weighted Clustering Algorithm) [CDT02] and **ZRP** (Zone Routing Protocol) [HP99]. Clustered routing is a vast field in itself and since it is not directly related to this work, we shall not delve into the specifics of these algorithms.

[2] http://www.pdos.lcs.mit.edu/grid

2.5 FACE ROUTING

FACE ROUTING evolved from the concept of the *right hand rule* [BM76] which states that each wall (edge) in the maze can be visited (traversed) while walking forward and with the right hand on the wall. This rule has been used to define the FACE ROUTING algorithm in [KSU99] and [BMSU99].

A line-of-sight segment is defined as the line segment connecting the source s and the destination t nodes (see Figure 2.6). In the following, we generally refer to the line segment between the source s and destination t as the $\overline{st}$ line. The $\overline{st}$ line intersects some faces of the planar graph.

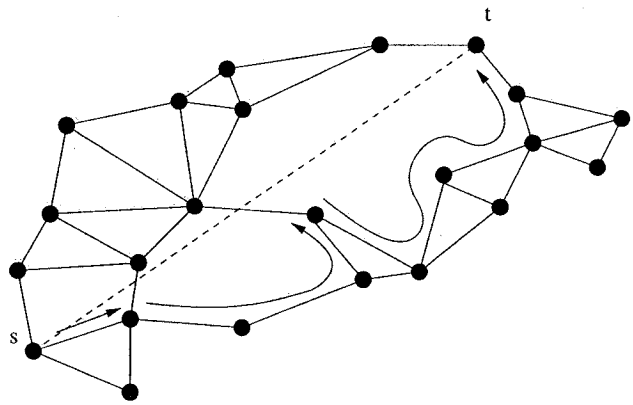


Figure 2.6: Face Routing.

FACE ROUTING starts at the source and traverses the first clockwise edge from the $\overline{st}$ line. The algorithm at each node continues to traverse an edge clockwise from the previously traversed edge until it reaches the target t or an edge cut by the $\overline{st}$ line. Each time the packet encounters an intersection that is closer than the previous such intersection, the packet is forwarded to the next face in the clockwise order (by skipping that edge and taking the following clockwise edge). This ensures progress. The version of FACE ROUTING we use in this thesis is given in Figure 2.7. Figure 2.6 illustrates the path taken by FACE ROUTING in an example graph. The packets always remain interior to a face cut by the $\overline{st}$ line. Thus the $\overline{st}$ line always acts as a reference and is required to ensure progress. FACE ROUTING thus guarantees a 100% delivery rate (*Theorem 5* of [BMSU99]) provided the underlying graph is planar.

FACE ROUTING **Algorithm**

Input: A planar geometric graph G,
a source vertex s and a target vertex t.

1. $c \leftarrow s$; // set the current vertex to be the source.
2. $distance \leftarrow dist(s,t)$;
3. $v \leftarrow$ the end-point of the first clockwise edge from $\overline{st}$ incident with s;
   // We follow the face of G containing the initial part of $\overline{st}$
4. Looping $\leftarrow$ false; // initialization
5. First_Edge$\leftarrow [s,v]$; //the first edge of a face, (ordered pair)
6. while $(v <> t)$ and not Looping do
   if ($[c,v]$ intersects $\overline{st}$ in point p) and $(dist(p,t) < distance)$
   then //switch to the adjacent face
   $v \leftarrow$ the end-point of the first clockwise edge from $[c,v]$ incident with c;
   First_Edge$\leftarrow [c,v]$; //the first edge of a face, (ordered pair)
   $distance \leftarrow dist(p,t)$; //remaining distance to the target
   else // continue with next edge along the current face
   $z \leftarrow$ the end-point of the first clockwise edge from $[c,v]$ incident with v;
   $c \leftarrow v$;
   $v \leftarrow z$;
   if First_Edge$= [c,v]$ then Looping $\leftarrow$ true;
   endwhile;
7. if Looping //we run around a face without progressing to the target
   print "there is no route from s to t";

Figure 2.7: The FACE ROUTING algorithm.

The FACE ROUTING algorithm described here is based on the *face-2* algorithm given by [BMSU99]. The original FACE ROUTING algorithm given by [KSU99] achieves the same effect, but by traversing each face completely and selecting the face-change point closest to the destination. Although face-1 achieves selection of the best point to change faces, a price is paid as it has to store information (in the packet header) about all the intersections of the edges of the current face with the $\overline{st}$ line.

Chapter 3

A New Algorithm for Routing in Some Non-Planar Networks

In this chapter, we propose our extensions to the FACE ROUTING algorithm in [BMSU99] to enable routing in non-planar graphs with a limited number of crossings. In Section 3.1, we start with the description of the network model including the motivating issues for this work and a definition of the types of non-planar graphs we work with. In the next section (Section 3.2), we give a description of our algorithm FRONC (**Face Routing On Networks with Crossings**), along with a justification of its actions and proof of its correctness. We illustrate the working of FRONC in Section 3.3 with some examples and discuss a couple of limitations in Section 3.4.

3.1 The Network Model

The following are the specifications of a commonly used model of an ad hoc wireless network.

- Nodes/hosts are randomly and uniformly distributed on a Euclidean plane.
- Each host has an omnidirectional antenna.
- All communication links are *bidirectional*, i.e., if a host u is able to receive signals from a mobile host v, then v can also receive signals from u. Note that this can be enforced even in cases where communication ranges of hosts are different or irregular (see [BFNO03])

- Each host can transmit with a constant power 'r' - denoted by the radius of each node's radio range. This forms the unit distance for a unit disk graph.
- The network is represented by a geometric undirected graph $G = (V, E)$, with vertices representing mobile hosts, and an edge connecting any pair of hosts that can communicate directly.
- Each host knows it's exact location in the plane in the form of x and y co-ordinates. The source of a unicast routing packet is assumed to know the location of the destination.

The last specification, viz; each node knowing its exact co-ordinates in the plane can be obtained by the use of a GPS system.

The above is the unit disk graph model for an ad hoc network. However, as discussed in Section 2.2.1, in the presence of obstacles, or irregular transmission ranges, a unit disk graph model may not be accurate for an ad hoc network. In these situations, some links that are present in the UDG representation are not actually present in the network. This may cause the planarization algorithm to produce a graph that does have crossings, which in turn may cause FACE ROUTING to fail. So, it is important to consider generalizations or variations of the unit disk model. In this thesis, we consider graphs derived from planar graphs by adding a limited number of crossings. We summarize below some of our reasoning for looking at these graphs.

- It is sometimes possible for a planarization algorithm to fail in planarizing a graph in the presence of obstacles. In this case, the underlying graph is not a unit disk graph. As shown in Figure 3.1, the nodes u and w should be connected in the unit disk graph model, but the link is not present because of an obstacle. So, node w retains the edge (x, w) even though u is present inside the disk of diameter $d(x, w)$. w is not able to avoid this due to the existence of the obstacle.
- Non-uniform radio ranges of nodes in a network complicate planarization.
- As will be discussed later (Section 3.2.1), the only link to the destination could be hidden inside a face.

Planar Graphs with Disjoint Crossings Added

We define *a planar geometric graph with k disjoint crossings added* to be:

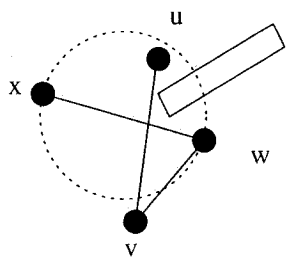


Figure 3.1: w does not remove the edge (x,w) during planarization.

$G_k = G \cup \{e_1, e_2, e_3, \ldots e_k\}$ where,

- G is a planar geometric graph.
- for every i, e_i crosses *exactly one* edge e_i^c in G.
- The faces of G containing e_i are distinct from the faces of G containing e_j when $i \neq j$, for $1 \leq i, j \leq k$.

3.2 The Algorithm FRONC

In this section, we give a description of our algorithm FRONC for planar graphs with disjoint crossings added, along with a justification of the correctness of its actions. For the sake of simplicity, we consider the case of $G_1 = G \cup \{e_1\}$. There are two different situations that need to be addressed by FRONC . Either G and $G \cup e_1 - e_1^c$ are both connected (we call this 'Case 1'), or one of them is not connected (we call this 'Case 2').

3.2.1 Algorithm FRONC Explained

Case 1

Case 1 investigates the situation where G and $G \cup e_1 - e_1^c$ are both connected. There exist many instances of the routing problem where FACE ROUTING does not encounter either of the edges e_1 or e_1^c and thus finds a route to the destination since G and $G \cup e_1 - e_1^c$ are planar. Thus we only need discuss how to find a route when FACE ROUTING encounters both e_1 and e_1^c. FRONC keeps a running list of edges of the face traversed so far (with the direction of traversal) called *edge-list*. FRONC selects the next edge to be traversed exactly as in

FACE ROUTING so long as the next edge selected either does not cross one of the edges in *edge-list* or it crosses one of the edges in *edge-list* and also the $\overline{st}$ line at a point closer to the target . However, if the current vertex in the routing is c and the next edge $[c,v]$ as determined by FACE ROUTING crosses an edge $[v_1,v_2]$ in *edge-list* then the next edge taken by our algorithm is selected as follows: Let x be the intersection of the edges $[v_1,v_2]$ and $[c,v]$. If going clockwise from v_1 around x we encounter c before v then FRONC continues with the first clockwise edge, say $[c,z]$, from the edge $[c,v]$ (see Figure 3.2 (a)). If instead we encounter v before c then we continue with the first clockwise edge, say $[v,z]$, from the edge $[c,v]$ (see Figure 3.2(b)).

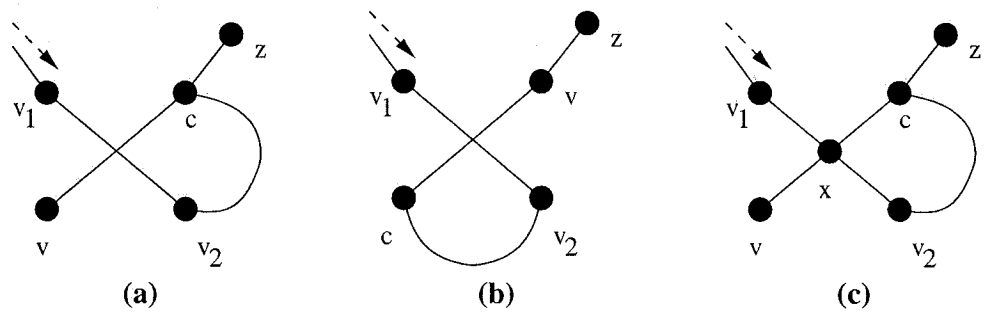


Figure 3.2: Configurations of crossing edges in case G is connected.

To justify the correctness of this action consider the geometric graph G_2 which is obtained from G_1 by placing a new vertex x at the position of the intersection of the edges e_1 and e_1^c and by replacing $e_1 = [v_1,v_2]$ and $e_1^c = [c,v]$ by $[v_1,x]$, $[x,v_2]$ and $[c,x]$, $[x,v]$, Clearly, G_2 is planar and if we run the FACE ROUTING in G_2, from vertex v_1 we take the edge $[v_1,x]$ and then either to $[x,c]$ followed by the edge $[c,z]$ or to $[x,v]$ followed by the edge $[v,z]$ (see Figure 3.2(c)). Since the decision of FACE ROUTING in G_2 leads to a route in G_2, the decision of FRONC in G_1 leads to a route from s to t.

Case 2

Case 2 investigates the case where G is not connected as in Figure 3.3 (the case when $G \cup \{e_1\} - \{e_1^c\}$ is not connected is identical and can be derived by exchanging the roles of e_1 and e_1^c). Since G is planar and G_1 is connected, G consists of two components C_1 and C_2. Geometrically, there are basically two possible configurations: In the first one C_1 is in

the outer face f of C_2 and vice-versa and the edge e_1 has one end-vertex on the outer-face of C_2 and the other end-vertex in on an internal face f_1 incident with the outer face of C_1 (see Figure 3.3(a)). The second configuration is that C_2 is inside one of the internal faces f of C_1 and the edge e_1 has one end-vertex on the outer-face of C_2 and the other end-vertex in on an internal face f_1 of C_1 adjacent to f (see Figure 3.3(b)).

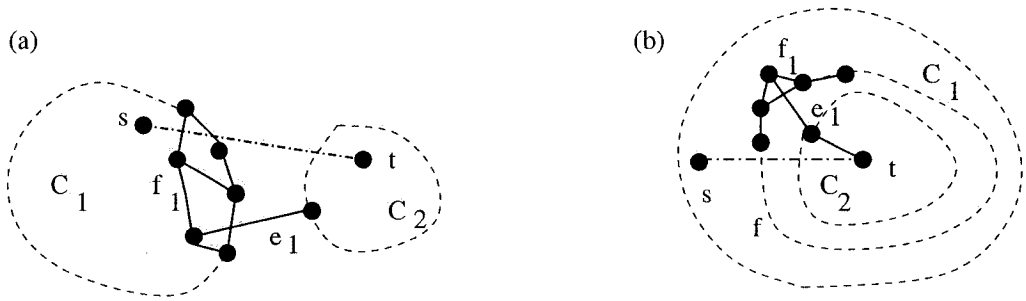


Figure 3.3: G is not connected

If both source and destination are in the same component, then one can observe that FRONC succeeds in routing based on the actions described in the previous case. Similarly if $\overline{st}$ intersects the face f_1 containing the edge e_1, FACE ROUTING eventually reaches the face f_1. In that case FRONC detects the crossing edges. If the source is in C_2 and the target in C_1 then FACE ROUTING will eventually find the edge e_1 Once the routing reaches C_1, we are back to the previous case with one crossing and our algorithm can find the remaining part of the route. Thus it remains to discuss the routing when the source is in C_1 and the target is in C_2. Notice that in this case any route from s to t contains the edge e_1. Since the action corresponds to routing in a planar graph equivalent to a graph in which the crossing point is replaced by a vertex, the correct edge is taken.

However, if the line segment from s to t does not intersect the face f_1 then FACE ROUTING ends up in a loop corresponding to the traversal of the outer face f_2 of C_1. Since we know f_1 is necessarily a face adjacent to f_2, FRONC systematically examines all the faces adjacent to f_2 looking for an edge crossing f_2. Once the crossing edge is found, we continue with the edge selected as in the first case. Since this moves the routing to C_2 and C_2 is planar, the algorithm finds a route to the target.

The two cases described above are combined to produce the algorithm FRONC . Although in the above algorithm we discussed the case of one crossing edge, clearly once a

Input: A geometric graph G_1 containing a pair of crossing edges,
a source vertex s and a target vertex t.

1. $c \leftarrow s$; // set the current vertex to be the source.
2. $distance \leftarrow dist(s,t)$;
3. $v \leftarrow$ the end-point of the first clockwise edge from $\overline{st}$ incident with s;
   // We follow the face of G containing the initial part of the $\overline{st}$ line segment
4. Looping = false; // initialization
5. First_Edge$\leftarrow [s,v]$; //the first edge of a face, (ordered pair)
6. Edge_List $\leftarrow [s,v]$; initialize the edgelist
7. while $(v <> t)$ and not Looping do
   if ($[c,v]$ intersects $\overline{st}$ in point p) and $(dist(p,t) < distance)$
   then //switch to the adjacent face
   $v \leftarrow$ the end-point of the first clockwise edge from $[c,v]$ incident with c;
   First_Edge$\leftarrow [c,v]$; //the first edge of a face, (ordered pair)
   Edge_List $\leftarrow [c,v]$; initialize the edge list
   $distance \leftarrow dist(p,t)$; //remaining distance to the target
   else // continue with next edge along the current face
   $z \leftarrow$ the end-point of the first clockwise edge from $[c,v]$ incident with v;
   if $[v,z]$ does not intersect any edge in the Edge_list then
   $c \leftarrow v$;
   $v \leftarrow z$;
   Append $[c,v]$ to Edge_List
   else // intersects edge $[v_1,v_2]$ of the list
   if c is found first clockwise from edge $[v_1,v_2]$ around v_2
   then $v \leftarrow$ the end of the first clockwise edge from $[c,v]$ around c;
   else $z \leftarrow$ the end of the first clockwise edge from $[c,v]$ around v;
   $c \leftarrow v$;
   $v \leftarrow z$;
   if (First_Edge= $[c,v]$) then //went around a face without progressing to the target
   search the faces adjacent to the edge-list for an intersecting edge;
   if (intersecting edge $[v_1,v_2]$ is found)
   $c \leftarrow v_1$ where v_1 is the vertex incident with an adjacent face;
   $v \leftarrow v_2$
   else (looping $\leftarrow$ true;
   endwhile;

Figure 3.4: Algorithm FRONC

crossing edge is resolved, the algorithm can deal with another crossing edge. The pseudocode for our algorithm FRONC is given in Figure 3.4.

3.2.2 Proof of Correctness of the Algorithm

Theorem

Let $G_k = G \cup \{e_1, e_2, e_3, \ldots e_k\}$ be a planar geometric graph with k disjoint crossings added. If G_k is connected then FRONC finds a route between any pair of vertices of G_k. At any point in the routing, the algorithm selects exactly one vertex as the next one on the route using only information about the edges incident with the current vertex and an edge-list of length $O(l)$, where l is the maximum number of edges in any face in G.

Proof

The fact that FRONC finds a route between any pair of vertices of G_1 in the manner specified in the theorem follows from the the description of FRONC above and the discussion of the possible cases encountered by FRONC when routing. Due to the disjoint nature of the crossings in G_k, when a crossing edge is a part of the route followed by FRONC , either the crossing is resolved or FRONC finds an edge crossing $\overline{st}$ at a point closer to the destination. In that case the routing switches to another face of G, and the fact of using a crossing edge previously is not relevant anymore. Thus FRONC can deal with another possible crossing on the route later on. FRONC needs to keep track of the edges in the face that it is currently traversing. Once a face change is made, information about previous faces' edges is not required as a crossing, if it exists, can belong only to the current face. Thus FRONC requires a maximum memory of the size of the largest face in the graph with one of the edges in every pair of crossing edges removed. □

3.3 Algorithm FRONC Illustrated with Examples

Case 1

As discussed earlier, there might exist a number of instances where FACE ROUTING does not encounter either e_1 or e_1^c and thus finds a route to the destination as the graphs G and $G \cup e_1 - e_1^c$ are planar. Thus we only need to consider the graphs where the algorithm encounters both e_1 and e_1^c. One such example where FACE ROUTING fails is shown in Figure 3.5.

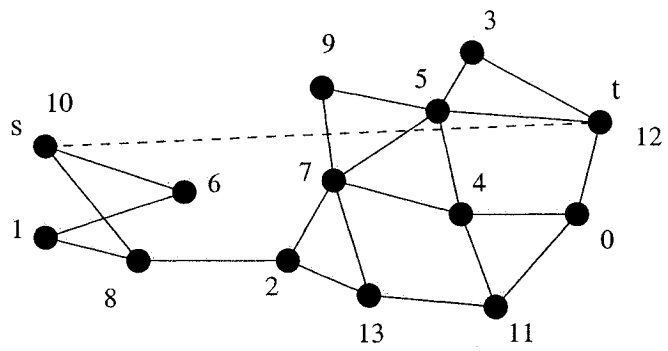


Figure 3.5: A non-planar graph where FACE ROUTING fails.

Referring to Figure 3.5, the sequence of events carried out by FACE ROUTING will be:

- Starting at source node 10, FACE ROUTING traverses the first clockwise edge from the $\overline{st}$ line, $(10,6)$.
- At node 6, FACE ROUTING picks the next clockwise node from 10 as 1 and traverses the edge $(6,1)$.
- At node 1, FACE ROUTING picks the next clockwise node from 6 as 8 and traverses the edge $(1,8)$.
- At node 8, FACE ROUTING picks the next clockwise node from 1 as 10 and traverses the edge $(8,10)$.
- At node 10, FACE ROUTING discovers that it is in a loop and stops.

In FRONC , we skip the edge $(8,10)$ and take the next clockwise edge when we discover that the path crosses over the path already traversed. An illustration on how FRONC works follows.

- Starting at source node 10, FRONC traverses the first clockwise edge from the $\overline{st}$ line, $(10,6)$.
- At node 6, FRONC picks the next clockwise node from 10 as 1 and traverses the edge $(6,1)$.
- At node 1, FRONC picks the next clockwise node from 6 as 8 and traverses the edge $(1,8)$.

- At node 8, FACE ROUTING picks the next clockwise node from 1 as 10. It discovers that it crosses the path already traversed (by checking in the *edge-list*. FRONC skips that node and takes the next following clockwise node 2 and traverses the edge $(8,2)$.

- At node 2, FRONC picks the next clockwise node from 2 as 7 and traverses the edge $(2,7)$.

- At node 7, FRONC picks the next clockwise node from 2 as 9. It discovers that it crosses the $\overline{st}$ line, say at point p. The $\overline{st}$ line is updated to this shortened length $\overline{pt}$. FRONC skips that node and picks the next following clockwise node 5. Again, it crosses $\overline{pt}$. p is updated to this shortened length. FRONC skips 5 and picks the next following clockwise node 4 and traverses the edge $(7,4)$.

- At node 4, FRONC picks the next clockwise node from 7 as 5. It discovers that it crosses $\overline{pt}$. It updated p and picks the next clockwise node 0 and traverses the edge $(4,0)$.

- 0 is a neighbor of 12, the destination. It delivers the packet to 12.

Case 2

In Case 2 we consider situations where G is not connected. Examples of the two situations (C_1 and C_2 on outer face of each other and C_2 inside an interior face of C_1) are shown in Figure 3.6 (a) and (b).

Referring to Figure 3.6(a), the sequence of events carried out by FACE ROUTING will be

- Starting at source node 25, FACE ROUTING traverses the first clockwise edge from the $\overline{st}$ line, $(25,12)$.

- At node 12, FACE ROUTING picks the next clockwise node from 25 as 8. Since this crosses $\overline{st}$, p is updated and the next clockwise node is chosen. FACE ROUTING chooses 21. Since this cross $\overline{pt}$ again, p is updated and the next clockwise node is chosen. FACE ROUTING picks 11 and traverses the edge $(12,11)$.

- At node 11, FACE ROUTING picks the next clockwise node from 12 as 21. As this crosses $\overline{pt}$, FACE ROUTING updates p and moves over to choose the next clockwise node. FACE ROUTING picks 10 and traverses the edge $(11,10)$.

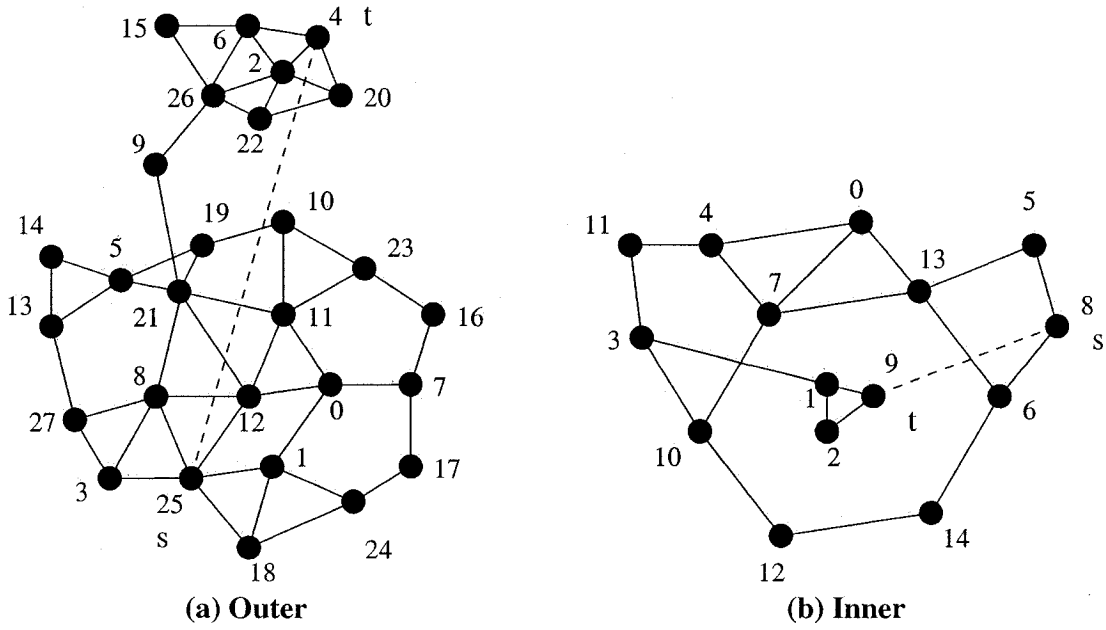


Figure 3.6: When edge (u,v) is removed, the graph is divided into two components.

- At node 10, FACE ROUTING picks the next clockwise node from 11 as 19. Since this crosses $\overline{pt}$, FACE ROUTING updates p and picks the next clockwise node 23 and traverses the edge $(10,23)$.
- FACE ROUTING traverses edges along this face (the perimeter) in the order $(23,16),(16,7),\ldots(19,10)$. FACE ROUTING then discovers that it is in a loop and stops.

FRONC starts to explore all adjacent faces of this face (the perimeter). FRONC uses the *left-hand rule*, a corollary of the right-hand rule. The following are the sequence of events after FACE ROUTING discovers that it is in a loop.

- FRONC explores face $(10,23,11)$. At 23, FRONC picks the next anti-clockwise node from 10 as 11. At 11, FRONC takes the next anti-clockwise node from 23 as 10. Now that FRONC is back on the perimeter, FRONC traverses the edges to explore the next face. Thus FRONC moves from $(10,23)$ to $(23,16)$.
- FRONC starts exploring the next face in a similar manner, eventually reaching the face $(5,21,19)$.

- At 19, FRONC picks the next anti-clockwise node from 5 as 21. At 21, FRONC then picks the next anti-clockwise node from 19 as 9. FRONC discovers that this crosses the path already traversed by checking the *edge-list*. FRONC thus traverses $(21,9)$.

- At 9, FRONC picks 26 as the next clockwise node. FRONC then traverses the perimeter of that component in the order $(9,26),(26,15),ldots(6,4)$, which is the destination.

A similar approach is used for the case where C_2 is inside an interior face of C_1. FACE ROUTING ends up in a loop on the face $(7,13,6,14,12,10)$ - see Figure 3.6(b). By traversing adjacent faces, FRONC discovers the edge $(3,1)$ to reach C_2 and eventually reaches the destination.

3.4 Limitations of FRONC

FRONC finds a route correctly in many cases when the added crossings are not necessarily disjoint as confirmed by our experiments (see Section 4.2). However, there are graphs with crossings added where FRONC fails for some source-destination pairs. One such graph is a graph consisting of two planar components say C_1 and C_2 and a path p. C_1 is in the outer face of C_2 and vice versa and path p that has one end-point at a vertex of C_1 that is close to the geographical center of C_1, crosses more than one face of C_1, and has the other end-point in C_2. FRONC fails to find a route from vertices in C_1 to C_2 if the $\overline{st}$ line does not cross the face of C_1 containing the attachment of path p in C_1. Figures 3.7(a) and (b) are examples of such graphs. In Figure 3.7(a), FRONC will not discover the crossing as the only accessible route is through the node x which is hidden in a face more than one layer inside the perimeter. This (the false perimeter) appears to be so as FRONC will traverse along (u,v) considering it to be the perimeter of C_1. However, the occurrence of such a situation is very rare.

Another example where FRONC fails to discover a route to the destination is shown in Figure 3.7 (c). FRONC will end up in a loop on the edges (s,u,v,w,x,y,z,s). Note that the examples shown here where FRONC fails are only in the case where the crossing edges are not disjoint.

Another limitation of FRONC is that it requires a memory of $O()$, where k is the size of the maximum number of edges in any face of the graph obtained by removing one edge in

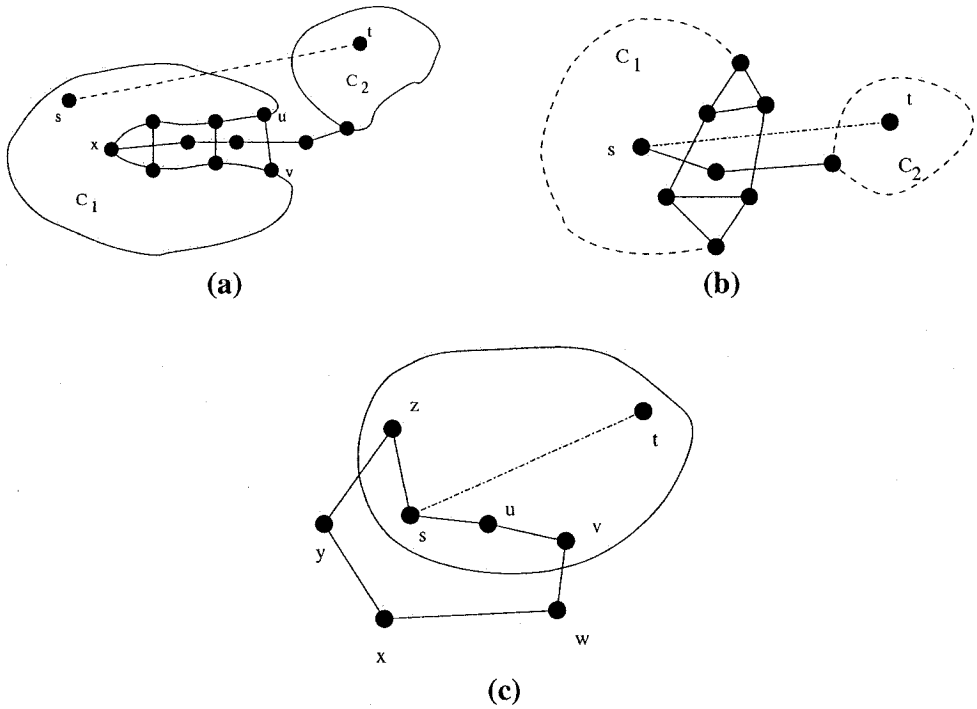


Figure 3.7: FRONC fails to discover the route when crossings are not disjoint.

each pair of crossing edges. Recall that FACE ROUTING can be considered a memoryless algorithm, since at each step, the algorithm makes the decision of the next edge solely on the basis of the edge most recently traversed. In contrast, FRONC needs to keep track of the edges in the face that it is currently traversing. Once a face change is made, information about previous faces' edges is not required as a crossing, if it exists, can belong only to the current face. Thus FRONC requires a maximum memory of the size of the largest face in the graph with one of the edges in every pair of crossing edges removed.

Chapter 4

Simulation Results

In this chapter we discuss the experiments that we conducted to study the performance of FRONC and FACE ROUTING and the results we obtained. The two performance measures we use are the *delivery rate*, defined as the percentage of times a route is successfully discovered, and the *stretch factor*, defined as the ratio of number of hops in the path calculated by a routing algorithm to the shortest path available.

4.1 Simulation Setup

Two types of simulations have been done in this thesis. In the first type, we create an input graph by generating a unit disk graph, extracting its Gabriel subgraph, and then adding some crossings to this graph. In the second type, we generate a graph corresponding to a network where nodes have irregular transmission ranges, and then run the Gabriel graph algorithm on it to get a graph which has crossings.

4.1.1 Gabriel Subgraphs of Unit Disk Graphs with Crossings Added

Recall that there are two cases possible for such graphs. To generate graphs conforming to the first case, the following specifications are used to generate the graph. The area of simulation is 500 x 500 units. Vertices are generated using a random uniform distribution. The transmission radius is 100 units. This is used to obtain a unit disk graph. Disconnected graphs are discarded. The graph is then planarized to obtain the Gabriel graph. With a random source and destination chosen, the path from s to t is computed using FACE ROUTING . Along this path, an edge is added between two vertices which are within

each other's transmission range if the added edge crosses an edge on the route, shown in Figure 4.1 is an edge (u,v) added which crosses (x,y). In this work, we add upto 3 such crossings. The number of nodes varies (75, 100 or 125) for different sets of experiments. A total of 1000 such graphs are generated for each combination of the above parameters; each tabulated result is the average computed for 1000 graphs. We also generate unit disk graphs and run the two algorithms on them without planarization.

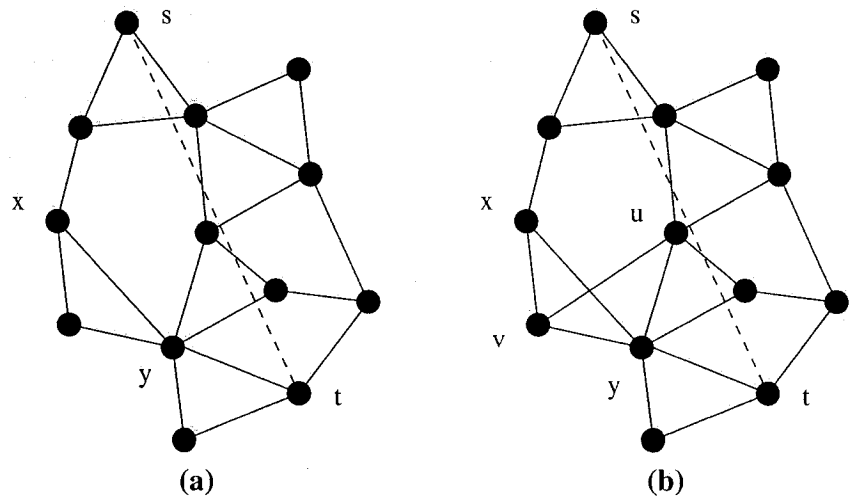


Figure 4.1: Construction of non-planar graph (Case 1).

For the second case, where C_1 and C_2 are on the outer face of each other, we generate a component of 60 vertices (C_1) in a square area of 300 x 300 units in a lower left corner of the 500 x 500 simulation area. We create another component (C_2) of 10 vertices in a 75 x 75 unit area in the upper right part of the square. These vertices are generated using a uniform random distribution. The transmission range is again 100 units. The unit disk graph is computed and the graph is planarized by applying the Gabriel graph construction algorithm. The source s is randomly selected from C_1. The destination t is randomly selected from C_2. Selecting the node from C_1 which connects to C_2 is crucial: the node is present in a face adjacent to the outer face but does not lie on the perimeter. The selected node is also ensured *not* to be present in a face cut by the $\overline{st}$ line. A chain of vertices is created to connect the selected node to a node on the perimeter of C_2.

To create the case where C_2 is inside an internal face of C_1, we generate 90 vertices in the the area depicted by region 'C' in Figure 4.2. Again, the simulation area is 500 x 500

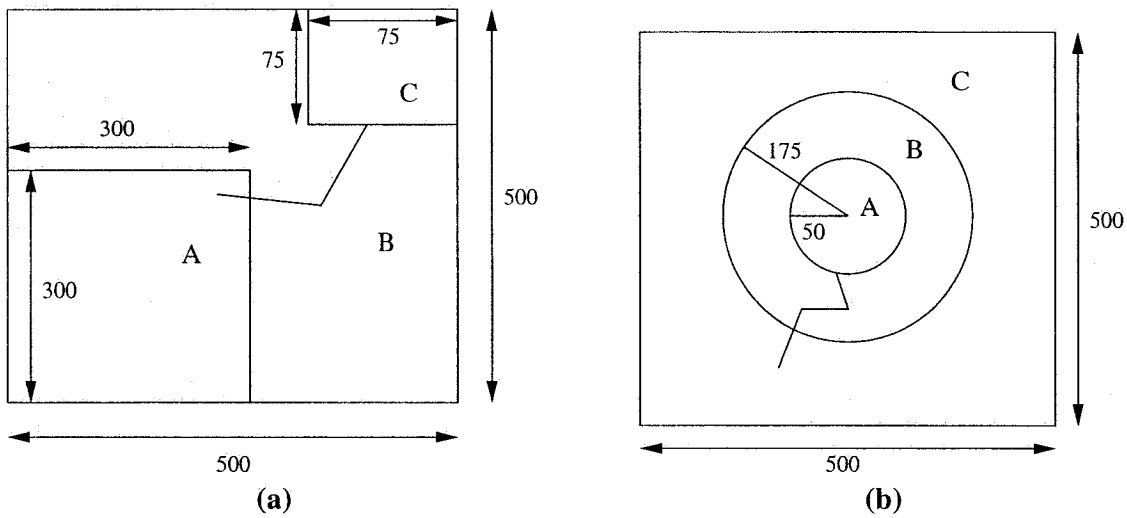


Figure 4.2: Construction of non-planar graph (Case 2).

units. A central area enclosed by a circle of radius 50 units (region 'A') is reserved for 5 nodes of component C_2. The transmission range is 100 units and is used to obtain a unit disk graph, which is planarized using the Gabriel graph algorithm. 5 nodes are generated in the region between the two components to connect C_1 and C_2 (region 'B'). The selected node in C_1 which connects to C_2 is ensured to belong to a face adjacent to the face enclosing C_2. Again, the selected node is ensured *not* to be present in a face cut by the $\overline{st}$ line.

4.1.2 Graphs Modeling Networks of Nodes with Irregular Transmission Ranges

As discussed in Section 2.2.1, the unit disk graph model is based on the assumption that the transmission range of a terminal is a perfect disk. The model described in [BFNO03] has terminals with irregular transmission ranges. A comparison of the two models is shown in Figure 4.3.

In our simulation, we generate the nodes as before, but to simulate the model shown in Figure 4.3(b), we use two parameters r and R. All nodes within distance r from a particular node v can be communicated with directly. For all nodes present farther than distance R, there can be no direct communication. For all nodes in the region between r and R, there may or may not exist a link. The probability that a node at distance d, where $r < d \leq R$,

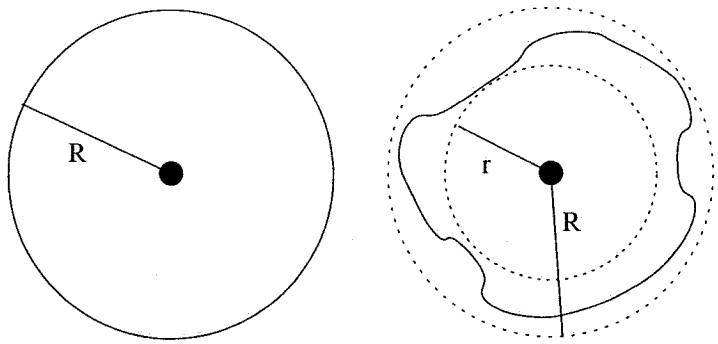


Figure 4.3: Comparison of the UDG model and a model with irregular transmission ranges.

is connected to v is denoted $P_{neighbor}$. This is shown in Figure 4.4. The Gabriel graph extraction algorithm is applied to the graph. In Figure 4.5, three different situations that are possible when applying the Gabriel graph algorithm to the edge (x,y) in our type graph are shown. In Figure 4.5(a), the node z which is present in the circle containing x and y with diameter dist(x,y) is visible to both x and y. This is the same as in the UDG model, and as in the case, the Gabriel graph test removes the edge (x,y). In Figures 4.5(b) and (c), x is not able to communicate with z, although it exists inside the circle. Removing the edge (x,y) could disconnect the graph, and therefore in our experiments, we retain such an edge. However, this may lead to crossings being present in the resulting graph.

We ran simulations with varying R/r values and varying $P_{neighbor}$ values. In all experiments, we have a constant value of $R = 100$ and we changed the value of r to get a specific value of R/r. We considered values of $R/r = 1, 1.2, 1.4, 1.6, 1.8, 2, 3, 4, 5$ and a value of $r = 0$, which translates to a model where all edges inside the disk formed by R are chosen with a probability $P_{neighbor}$. We experimented with several values of $P_{neighbor}$: $0.2, 0.4, 0.5, 0.6, 0.8$ and 1.

4.2 Results

In this section, we discuss the results of the simulations. As pointed out earlier, we are interested in discovering the delivery rate and the stretch factor.

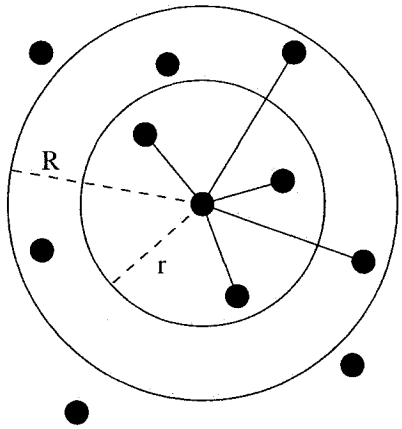


Figure 4.4: Generation of a graph based on the model given in [BFNO03].

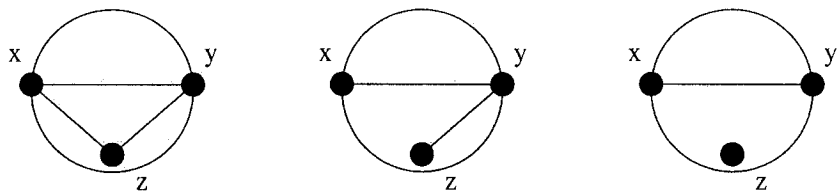


Figure 4.5: Applying the Gabriel graph test on the irregular transmission model can result in crossings.

4.2.1 Unit Disk Graph Model: Case 1

Figure 4.6 shows the effect of the number of vertices on the stretch factor for a fixed number of crossings, while Figure 4.7 shows the the effect of the number of crossings on the stretch factor for a fixed number of vertices.

As can be seen, FRONC has a smaller stretch factor as compared to FACE ROUTING in all the categories. The stretch factors decrease for both FACE ROUTING and FRONC as the graphs become denser. Note that the scale of the generated graph is different for Figures 4.6 (a), (b), (c) and (d). For the unit disk graph (Figure 4.6(d)), the stretch factors increase as the number of vertices increase. This may be due to the fact that the types and number of non-planarities are uncontrolled.

In Figure 4.7, we see the effect of number of crossings for a particular number of

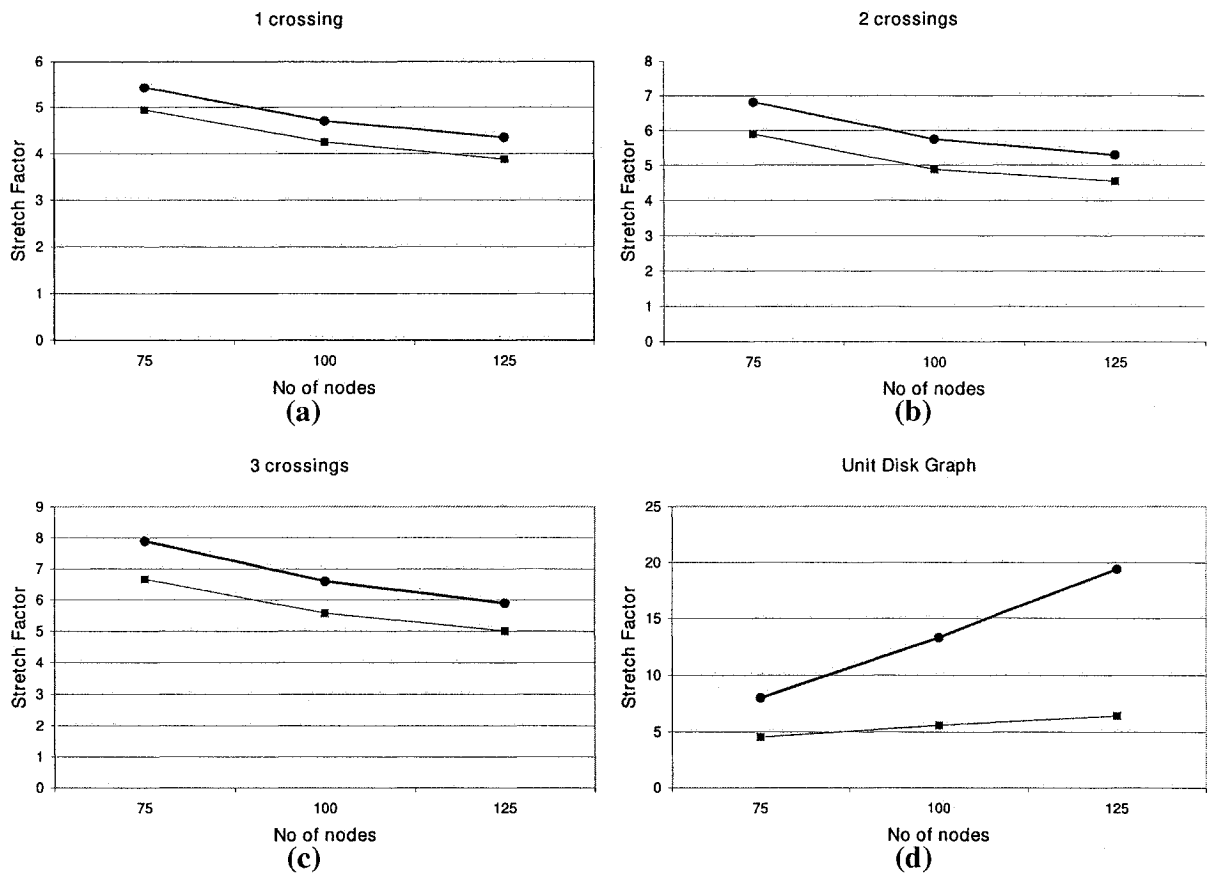


Figure 4.6: Effect of number of vertices for a particular number of crossings. FACE ROUTING , FRONC values are shown as ○, □ respectively.

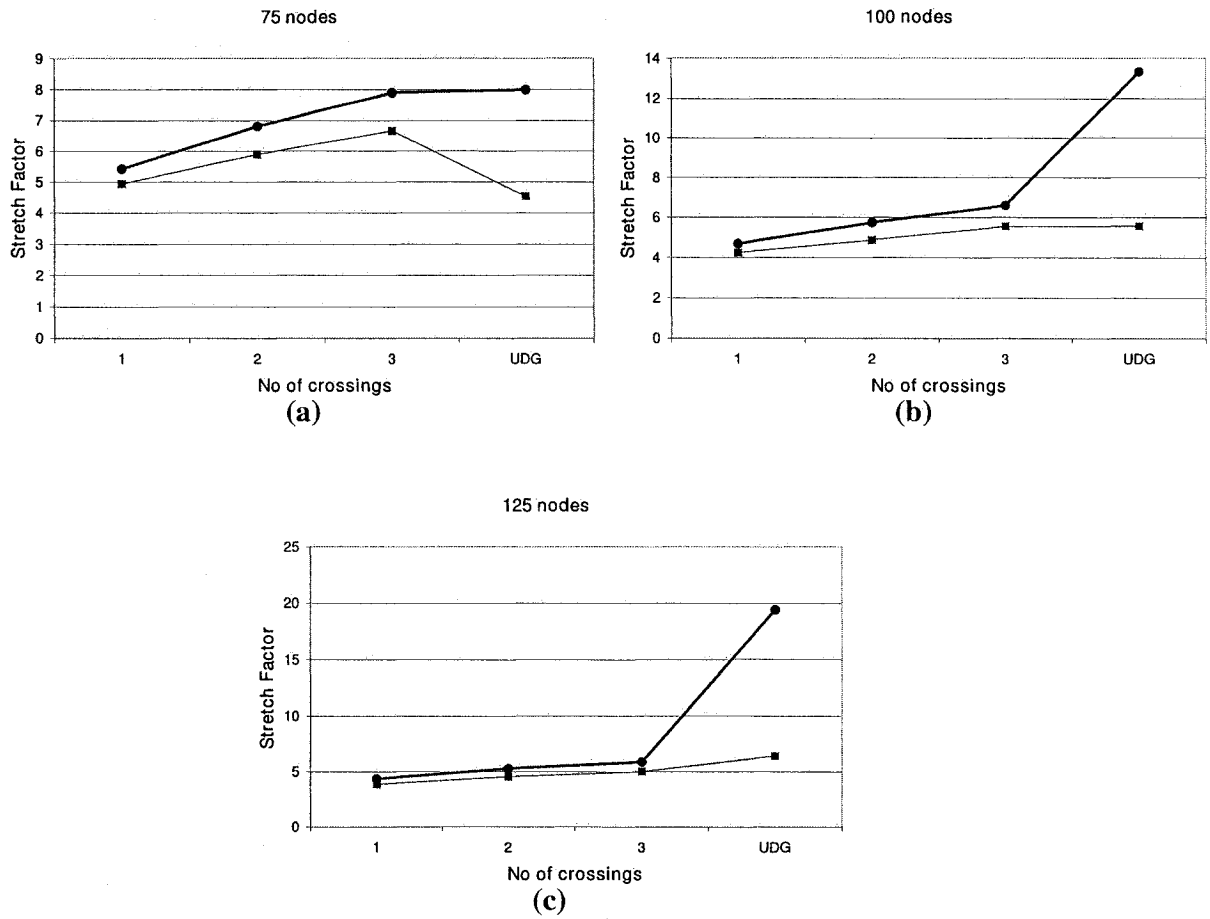


Figure 4.7: Effect of number of crossings for a particular number of vertices. The FACE ROUTING , FRONC values are shown as ○, □ respectively

vertices. As is apparent, the stretch factor increases exponentially for a UDG when the graphs are denser. Again, note that the scales are different.

Table 1: Delivery rates for FACE ROUTING and FRONC in Case 1

Face Routing			
No. of vertices	75	100	125
	Delivery rates		
1 crossing	99.1	99.6	99.6
2 crossings	99	99.3	99.6
3 crossings	98.5	99.1	99.1
UDG	99.7	99.3	99.8

FRONC			
No. of vertices	75	100	125
	Delivery rates		
1 crossing	100	100	100
2 crossings	100	100	100
3 crossings	100	100	100
UDG	100	100	100

Table 1 shows the delivery rate of FACE ROUTING and FRONC for different combinations of crossings and number of vertices. The last line labeled UDG is for the case when the algorithms are run on unit disk graphs without any planarization. Our algorithm succeeds in all cases whereas FACE ROUTING fails sometimes. It is interesting to note that both algorithms have a high delivery rate for UDG.

4.2.2 Unit Disk Graph Model: Case 2

The results discussed so far were for the first case where G is connected. Results for the second case are in Table 2. In this case, the delivery rate of FACE ROUTING is always 0, and the delivery rate of FRONC is 100%. FACE ROUTING always fails due to the fact that the crossing edge e_1 is present in a face which is *not* cut by the $\overline{st}$ line segment.

Table 2: Delivery rates and stretch factors for FACE ROUTING and FRONC in Case 2.

C_1 and C_2 in outer face of each other			C_2 inside an inner face of C_1		
	Face Routing	FRONC		Face Routing	FRONC
Delivery Rate	0	100	Delivery Rate	0	100
Stretch Factor	-	5.71	Stretch Factor	-	25.91

For the case where C_1 and C_2 are in the outer face of each other, the stretch factor is relatively low (see Table 2) because the crossing edge e_1 is added in one of the faces close

to the the face cut by the $\overline{st}$ line. In the case where C_2 is inside a face of C_1, the delivery rate is again 0 for FACE ROUTING and 100% for FRONC . The stretch factor is relatively high as the algorithm checks on all adjacent faces for a crossing edge and it could actually be on the outer face of C_1. This results in the algorithm traversing the outer face of the graph multiple times. Another fact that adds up to the stretch factor is that the vertex in C_1, which is an end-point of e_1 is chosen randomly.

4.2.3 Model with Irregular Transmission Ranges

The first set of simulations for this model were conducted to see the effect of varying $P_{neighbor}$ on the stretch factor and the effect of varying R/r on the stretch factor. The results are shown in Figure 4.8(a) and (b). The solid lines represent the results for 75 nodes, dotted lines represents 100 nodes and the dot-and-dashed lines represent 125 nodes. The circles and squares represent FACE ROUTING and FRONC respectively. The stretch factors for FRONC and FACE ROUTING are very similar though the stretch factor for FRONC is slightly better than that of FACE ROUTING in many cases.

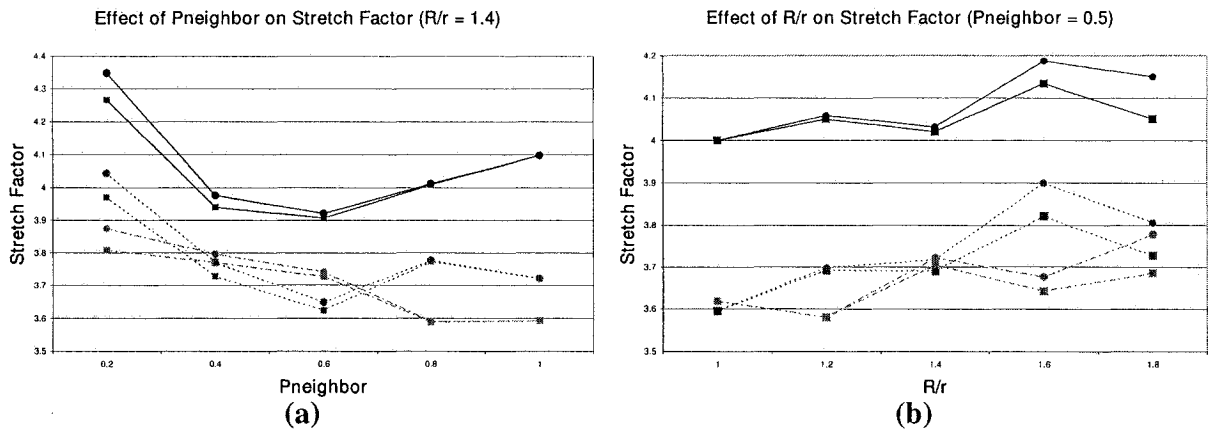


Figure 4.8: Simulation results for smaller values of R/r.

We noted from these experiments that with such low values of R/r, there were hardly any crossings in the graphs, which explains why the performance of the two algorithms is so similar. Table 3 shows the number of crossings per graph obtained from the simulations. As can be expected, the delivery rate for both algorithms in this set of experiments ($R/r < 2$)

is close to 100%.

Table 3: Number of crossings per graph for lower values of R/r.

$P_{neighbor} = 0.5$			
No. of vertices	75	100	125
$R/r = 1$	0	0	0
$R/r = 1.2$	0.036	0.044	0.045
$R/r = 1.4$	0.21	0.312	0.403
$R/r = 1.6$	0.612	0.924	1.093
$R/r = 1.8$	0.959	1.59	2.047

$R/r = 1.4$			
No. of vertices	75	100	125
$P_{neighbor} = 0.2$	0.671	1.026	1.319
$P_{neighbor} = 0.4$	0.33	0.464	0.555
$P_{neighbor} = 0.6$	0.129	0.196	0.242
$P_{neighbor} = 0.8$	0.032	0.051	0.058
$P_{neighbor} = 1$	0	0	0

To simulate a scenario where there exist more crossings in the graph, we increased the values of R/r for two values of $P_{neighbor}$ (0.2 and 0.5). Tables 4 and 5 give the delivery rates for the $P_{neighbor}$ values of 0.2 and 0.5. Figure 4.9(a) shows the number of crossings per graph for different values of R/r, with $P_{neighbor}$ set to 0.2. Figure 4.9(b) shows the effect of the varying R/r values on the stretch factors of FACE ROUTING and FRONC with $P_{neighbor}$ set to 0.2. Figure 4.10(a) shows the number of crossings per graph for different values of R/r, with $P_{neighbor}$ set to 0.5 and Figure 4.10(b) shows the effect of varying R/r values on the stretch factors of FACE ROUTING and FRONC with $P_{neighbor}$ set to 0.5. The circles and squares in Figures 4.9(b) and 4.10(b) represent FACE ROUTING and FRONC respectively. In both Figures 4.9 and 4.10, the solid lines represent 75 nodes, the dotted lines indicate 100 nodes and the dot-and-dashed lines indicate 125 nodes.

Table 4: Delivery rates for FACE ROUTING and FRONC when $P_{neighbor} = 0.2$

Face Routing			
No. of vertices	75	100	125
	Delivery rates		
R/r=1	100	100	100
R/r=2	98.7	99.6	99.7
R/r=3	97.1	98.9	98.5
R/r=4	97.2	98.3	97.6
R/r=5	95.1	97.4	98.7
r=0	84.5	90.1	92.7

FRONC			
No. of vertices	75	100	125
	Delivery rates		
R/r=1	100	100	100
R/r=2	99.8	100	100
R/r=3	99.4	99.6	99.9
R/r=4	99.6	99.7	99.5
R/r=5	99.4	99.3	99.8
r=0	93.5	94.9	95.9

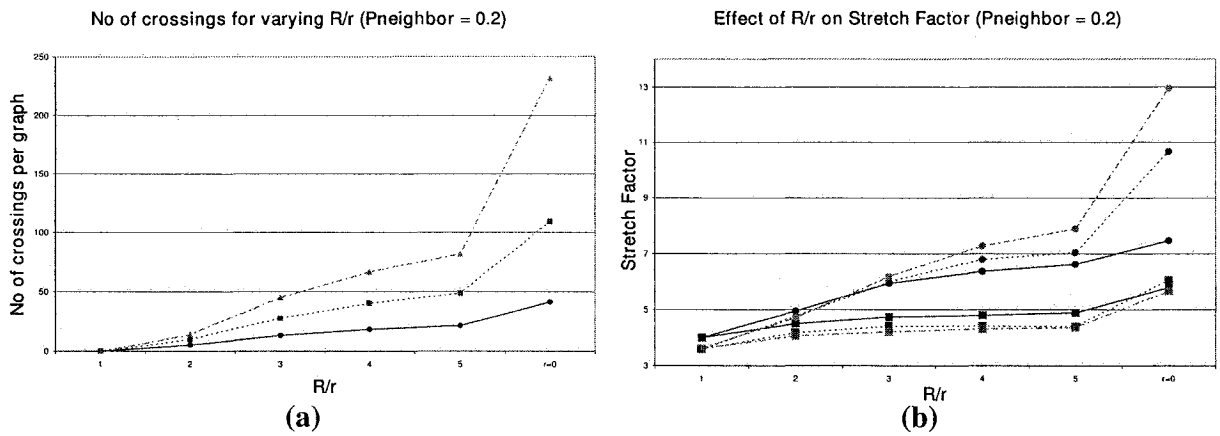


Figure 4.9: Simulation results for the case where $P_{neighbor}$=0.2.

It is clear that the number of crossings increases rapidly for higher values of R/r. The delivery rate of FRONC is always higher than that of FACE ROUTING and when $r = 0$, it is substantially higher. As can be noted, the stretch factor for FACE ROUTING is significantly higher for larger values of R/r. In contrast, the stretch factor of FRONC does not increase as dramatically. One can surmise that the high delivery rate of FACE ROUTING even in the presence of a large number of crossings is due to the fact that it succeeds in finding the route to the destination by wandering around in the graph. On the other hand, the experiments also suggest that FRONC deals with crossing edges by correcting for them, even when the crossing edges are not necessarily disjoint.

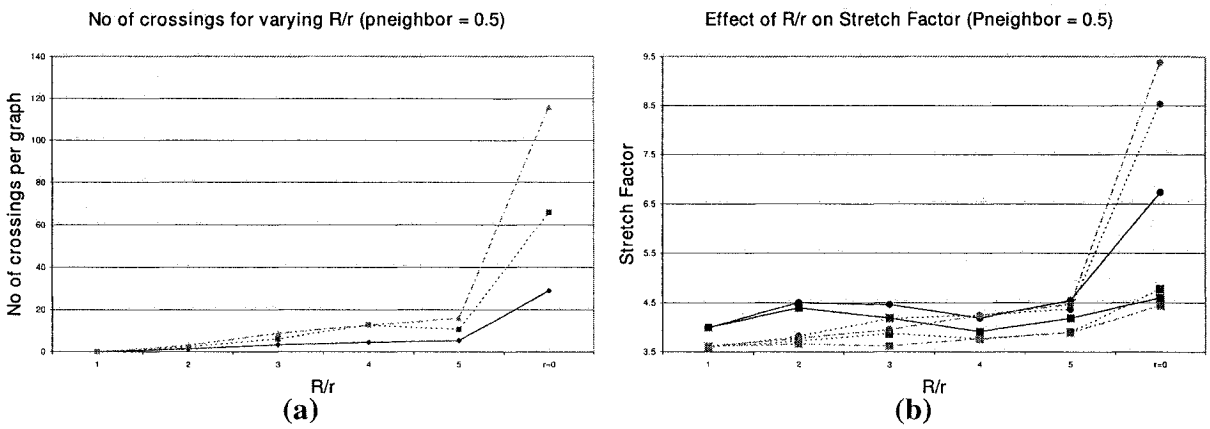


Figure 4.10: Simulation results for the case where $P_{neighbor}$=0.5.

Table 5: Delivery rates for FACE ROUTING and FRONC when $P_{neighbor}$ = 0.5

Face Routing			
No. of vertices	75	100	125
	Delivery rates		
R/r=1	100	100	100
R/r=2	99.7	100	100
R/r=3	99.6	99.8	99.9
R/r=4	99.4	99.9	99.9
R/r=5	99.5	99.6	99.9
r=0	95.8	96.7	98.2

FRONC			
No. of vertices	75	100	125
	Delivery rates		
R/r=1	100	100	100
R/r=2	100	100	100
R/r=3	99.9	100	100
R/r=4	100	100	100
R/r=5	99.9	100	100
r=0	98.7	99.4	99.6

Chapter 5

Conclusion

Mobile ad hoc networks are composed of autonomous and possibly heterogeneous wireless devices that communicate with each other over a radio link. In this work, we restricted our attention to routing in such networks with an emphasis on position-based routing, where the devices know their exact geographical location. In particular we looked at the FACE ROUTING algorithm which guarantees delivery provided the underlying graph is planar.

In this thesis, we investigate non-planar networks, with a limited number of disjoint crossing edges. We propose an extension to FACE ROUTING called FRONC which guarantees delivery in such networks. Essentially FRONC imitates the behavior of FACE ROUTING until it detects a crossing edge, at which point, it applies a local correction. If a loop is discovered, then FRONC examines the faces of the graph adjacent to the face on which there was a loop, to find and use the crossing edge. We prove the correctness of our algorithm as long as the crossing edges are disjoint. Our experiments show that our algorithm has a better stretch factor as compared to FACE ROUTING , and always guarantees delivery even when FACE ROUTING fails.

Routing in non-planar graphs is not a simple task, especially when only local information is available. We propose a first step towards guaranteeing delivery in networks based on non-planar graphs by investigating a particular class of non-planar graphs. We have guaranteed delivery in non-planar graphs when the crossing edges are disjoint. It is a worthwhile task to consider extensions of FACE ROUTING to more generic non-planar graphs such as graphs where the crossing edges are not disjoint. Another interesting problem would be to investigate graphs when no restriction is applied on the number of edges

that can be crossed by an edge. For Case 2, this is straight-forward, where the routing algorithm would explore n layers adjacent to the face where it detects a loop, where n is the number of edge-crossings allowed. Some research into this matter is required to see how this would affect Case 1.

One drawback of FRONC is that it requires $O(l)$ memory, where l is the maximum number of edges in any face in the graph obtained by removing one edge in each pair of crossing edges. Thus an interesting question that merits investigation is whether it is possible to design a memoryless routing algorithm for non-planar graphs.

Bibliography

[BBC+01] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J-P. Hubaux, and J-Y. Le Boudec. Self-organization in mobile ad hoc networks: the approach of terminodes. *IEEE Communication Magazine*, 39(6):166–175, June 2001.

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

[BFNO03] L. Barriere, P. Fraigniaud, L. Narayanan, and J. Opatrny. Position-based routing in wireless ad hoc networks with unstable transmission ranges. *Wireless Communications and Mobile Computing Journal*, 3/2:141–153, 2003.

[BM76] J.A. Bondy and U.S.R. Murty. *Graph Theory with applications*. Elsevier North Holland, 1976.

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

[BMSU99] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of 3rd ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM99)*, pages 48–55, August 1999.

[Car79] Bernard Carré. *Graph and Networks*. Clarendon Press, 1979.

[CDT02] M. Chatterjee, S.K. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *ournal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, 5(2):193–204, April 2002.

[CHH01] Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proc. of 34th Hawaii Internationl Conference on System Sciences*, 2001.

[Deo74] Narsingh Deo. *Graph theory with application to engineering and computer science*. Prentice-Hall, 1974.

[DM01] Douglas S.J. De Couto and Robert Morris. Location proxies and intermediate node forwarding for practical geographic forwarding. Technical report, MIT Laboratory for Computer Science, June 2001.

[DSW01] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, 2001.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[FL01] James A. Freebersyser and Barry Leiner. *Ad Hoc Networking*, chapter A DoD Perspective on Mobile Ad Hoc Networks, pages 29–51. Addison-Wesley, 2001.

[GS69] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

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

[HP99] Z.J. Haas and M.R. Pearlman. The zone routing protocol for ad hoc networks. Technical report, IETF Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999.

[JLT99] M. Jiang, J. Li, and Y.C. Tay. Cluster based routing protocol. Technical report, IETF Internet Draft, draft-ietf-manet-cbrp.txt, June 1999.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski, editor, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[Joh94] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the Workshop on Mobile Somputing Systems and Applications*, pages 158–163, December 1994.

[JPS01] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications Magazine*, 8(1):48–57, February 2001.

[KK00] B. Karp and H.T. Kung. Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, August 2000.

[KSU99] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of 11th Canadian Conference on Computational Geometry*, 1999.

[KV98] Y.B. Ko and N.H. Vaidya. Location aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM/IEEE Mobile Computing and Networking (MobiCom 98)*, 1998.

[KWZ02] F. Kuhn, R. Wattenhofer, and A. 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 (DIALM '02)*, September 2002.

[KWZ03] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc Networks Beyond Unit Disk Graphs. In *Proceedings of the 2003 joint workshop on foundations of mobile computing*, pages 69–78, 2003.

[KWZZ03] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proceedings of Principles of Distributed Computing - PODC 2003*, 2003.

[LJD+00] Jinyang Li, John Jannotti, Douglas S.J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.

[Mai04] Christian Maihöfer. A survey of geocast routing protocols. *IEEE Communications Surveys and Tutorials*, 6(2), April 2004.

[MC98] J.P. Macker and M.S. Corson. Mobile computing and communications review. Technical report, Mobile ad hoc networking and the IETF, 1998.

[MWH01] Martin Mauve, Jörg Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.

[NBN01] Navid Nikaein, Christian Bonnet, and Neda Nikaein. HARP - hybrid ad hoc routing protocol. In *Proceedings of IST 2001*, 2001.

[NI97] J. Navas and T. Imielinski. Geographic addressing and routing. In *Proceedings of 3rd ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 97)*, 1997.

[OTL04] R. Ogier, F. Templin, and M. Lewis. *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*. IETF RFC No. 3684, February 2004.

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

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

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

[Toh02] C-K. Toh. *Ad hoc mobile wireless networks: Protocols and Systems*. Prentice Hall PTR, 2002.

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