uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Hanna Kalosha

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Select and Protest based Beaconless Georouting with Guaranteed Delivery in Wireless Sensor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Amiya Nayak

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Ivan Stojmenovic

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Nejib Zaguia

Evangelos Kranakis

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Select and Protest based Beaconless Georouting with Guaranteed Delivery in Wireless Sensor Networks

by

Hanna Kalosha

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario K1N 6N5, Canada

Your file Votre référence
ISBN: 978-0-494-50892-3
Our file Notre référence
ISBN: 978-0-494-50892-3

Acknowledgements

I would like to acknowledge my sincerest gratitude to my supervisor Prof. Dr. Ivan Stojmenovic of Ottawa-Carleton Institute for Computer Science. Without his enthusiastic and expert guidance during writing this paper, it would not be possible to complete the thesis. I am grateful to my co-supervisor Prof. Dr. Amiya Nayak for the help in preparation of this thesis. I would like to thank Dr. Stefan Rührup for the valuable contribution and discussions of the subject and simulation results. I would also like to thank my family and my father Dr. Vladimir Kalosha in particular for their continuous understanding and support.

Contents

List of Figures

List of Tables

Abstract

Recently proposed beaconless georouting algorithms are fully reactive, with nodes forwarding packets without prior knowledge of their neighbors. However, existing approaches for recovery from local minima can either not guarantee delivery or they require the exchange of complete neighborhood information.

We describe two general methods which enable completely reactive face routing with guaranteed delivery. The Beaconless Forwarder Planarization (BFP) scheme finds correct edges of a local planar subgraph at the forwarder node without hearing from all neighbors. Face routing then continues properly. Angular Relaying determines directly the next hop of a face traversal. Both schemes are based on the Select and Protest principle. Neighbors respond according to a delay function, but only if there is no other neighbor within their forbidden region. Protest messages are used to correct occasionally wrong selections by neighbors that are not in the planar subgraph.

We show that a correct beaconless planar subgraph construction is not possible without protests. We also show the impact of the chosen planar subgraph construction on the message complexity. This leads to the definition of the Circlunar Neighborhood Graph (CNG), a new proximity graph that enables BFP with a bounded number of messages in the worst case, which is not possible when using the Gabriel graph (GG). The CNG is sparser than the GG, but this does not lead to performance degradation. Simulation results show similar message complexities in the average case when using CNG and GG.

Angular relaying uses a delay function that is based on the angular distance to the previous hop. We develop a theoretical framework for delay functions and show both theoretically and in simulations that with a function of angle and distance we can reduce the number of protests by a factor of 2 in comparison to a simple angle-based delay function.

Chapter 1 Introduction

1.1 Background

Wireless Sensor Networks are composed of a large number of heterogeneous nodes called sensors. The operation of sensors enables the distributed sensing of physical phenomenon. When a sensor detects an event it forms a packet to be delivered to one or a subset of selected information sinks, also known as actors. The sensor forwards the packet to a neighbor node selected according to an intelligent criteria and sends the packet to that node. The goal is to guarantee delivery of all packets so that no acknowledgements are necessary.

Geographic routing [10], [11], [16] forms a specific class of routing protocols which requires that each network node is able to determine its coordinates by means of a location system like GPS or relative positioning based on signal strength estimation [19]. Each routing step requires knowledge about the location of the message's final destination. When the destination location is not known in advance, it has to be requested by using a location service [28] which provides a mapping from node addresses to their physical locations.

The majority of geographic routing protocols enable message forwarding in a localized manner, i.e. deciding the next routing hop is based solely on a constant amount of information stored in the message, and the location of the current node, its neighbors, and the message's final destination. Localized routing protocols can further be classified regarding their delivery guarantees. Guaranteed delivery refers to the ability of successfully forwarding a message from source to destination. The definition requires that source and destination are connected by at least one path in the network and that we have an idealized MAC layer where messages are not lost during any forwarding step.

Elementary geographic routing algorithms employ the greedy routing principle by sending the message to the neighbor node which locally looks best regarding the destination position and the metric being optimized [31], [10]. For each localized greedy

routing variant the message may end up in a node that has to drop the message in order to prevent a routing loop. Dropping a message might be necessary even though a path exists from source to destination node. On the other hand, if successful the majority of greedy routing algorithms produce routing paths with a weight that is comparable to the weight of the shortest possible path. For this reason greedy routing is often applied in combination with a recovery strategy which is responsible for handling the message as long as greedy routing fails.

Planar graph routing which is also referred as face routing [4], [21], [23] is the most prominent recovery strategy preserving the stateless property of geographic greedy routing mechanisms. The basic idea is to planarize the network graph in a localized manner and to forward a message along one or possibly a sequence of adjacent faces which are providing progress towards the destination node.

Beaconless georouting algorithms follow the principle of geographic routing, where a message is routed to the location of the destination instead of a network address. It is based on the assumptions that each node can determine its own geographic position and that the source knows the geographic position of the destination. The use of position data enables routing without routing tables or prior route discovery. Conventional geographic routing algorithms use two basic forwarding principles: greedy forwarding and face routing. Greedy forwarding selects a neighboring node that minimizes the distance to the target. This strategy fails in case of a local minimum, i.e. if no neighboring node is closer to the destination. Then, face routing can be used in order to recover from this situation. During recovery the message is routed along the incident face of the communication graph using the right-hand rule until a position is found that is closer to the destination than the last local minimum. Face traversals work only on a planar subgraph, otherwise the crossing edges might cause a routing loop. Therefore, a local planarization strategy is needed which would determine the edges of a planar subgraph.

1.2 Existing Solutions and Motivation

1.2.1 Beaconless Routing

Conventional geographic routing algorithms rely on the position information of their 1-hop neighbors. This information can be gathered by a periodic exchange of beacon

messages. Beaconless routing algorithms attempt to avoid this message exchange and provide a completely reactive routing. This approach is also referred to as stateless position-based beaconless routing, location-aware contention-based forwarding or volunteer forwarding. The basic principle of beaconless forwarding is the following. The forwarder, i.e. the node that currently holds the packet, broadcasts it to its neighbors. The nodes within the forwarder's transmission range receive the packet, but only the nodes in the forwarding area are eligible for forwarding it further (see Fig. 1.1). The fundamental property of the forwarding area is that it consists of a set of sensor nodes that can mutually communicate with each other. The nodes within the forwarding area are called candidates. The most suitable candidate is determined by a contention mechanism: After receiving the packet, each candidate starts a timer. The timer is determined by a delay function that favors the most promising node, e.g., the node closest to the destination has the shortest timeout. This node forwards the packet again, when its timer expires. The other candidates notice that the packet is re-transmitted and cancel their timers. This strategy follows the greedy principle, because it always uses locally optimal decisions.

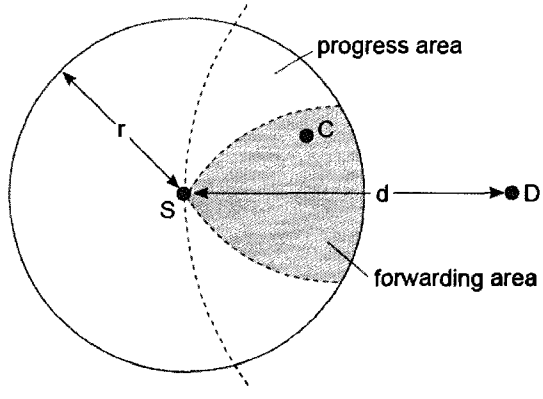


Fig. 1.1. Forwarder (S), candidate (C) and destination (D).

1.2.2 Beaconless Recovery Problem

As greedy routing fails in case of a local minimum, a recovery strategy is needed to provide guaranteed delivery. The preferred recovery method for conventional geographic routing algorithms is the face traversal on a planar subgraph, which is constructed from the neighborhood information. But for beaconless protocols the full knowledge of the neighborhood is not available a priori. Instead, a part of this knowledge has to be gained by exchanging messages, if it is not implicitly given by the location of the nodes.

1.3 Objectives

The first objective of this thesis is to introduce two novel solutions to the beaconless recovery problem and to combine known beaconless greedy routing with the proper beaconless recovery technique which would result in a routing algorithm with guaranteed delivery.

We can identify two questions related to the beaconless recovery problem, the answer to which is the key to guaranteed delivery:

1. How to construct a local planar subgraph on the fly?
2. How to determine the next edge of a planar subgraph traversal?

The beaconless recovery problem has to be solved reactively and with as few messages as possible. Existing approaches use a reactive message exchange in which all neighbors are involved in the worst case. This raises the question whether we can reduce this message overhead. Thus the second objective concerns achieving a significant message reduction in comparison to conventional protocols that rely on beaconing.

1.4 Contributions

In this thesis we answer the questions formulated in section 1.3 and provide new solutions for both variants of the beaconless recovery problem. We introduce the Beaconless Forwarder Planarization (BFP) which is a generalization of the recovery mode operation in the Guaranteed Delivery Beaconless Forwarding (GDBF) framework [5], [6] (see section 2.2.2.3). The BFP first constructs an approximation of the planar subgraph and then eliminates the nodes which are not neighbors in a planar subgraph. We use proximity graphs such as Gabriel graph and relative neighborhood graph for the planar subgraph construction because edges in these graphs can be determined locally. We propose the Circlunar Neighborhood Graph (CNG), a planar proximity graph that can be constructed with less messages than the Gabriel graph and that has a better connectivity than the relative neighborhood graph. The second novel solution of the beaconless recovery problem is Angular Relaying, which first tries to find the next neighbor of a right-hand face traversal and then switches to another neighbor, if the selected neighbor is not adjacent in the planar Gabriel subgraph.

To carry out the above mentioned experiments we have developed simulation software with the following functionality:

1. Random unit disc graph (UDG) generation with the given initial parameters (number of nodes, average density).
2. Greedy-face-greedy routing for the given number of random UDGs. In the recovery mode one of the following algorithms can be used: routing with Beaconless Forwarder Planarization (BFP) on GG, CNG or RNG; Angular Relaying on GG with sweep line; Angular Relaying on GG with semi-circle sweep curve.
3. Graphical and tabular representation of simulation results. Simulation results summary output.

This thesis is adapted from the paper [21] which has been accepted for publication in Proceedings of the 27th Conference on Computer Communications (IEEE INFOCOM 2008).

1.5 Assumptions

We assume that nodes are aware of their own position by means of GPS, or any other positioning service [19]. Furthermore, there is a mechanism that enables the source to detect accurately enough the destination node's position. But, opposed to conventional position-based routing algorithms described in Chapter 2, no beaconing mechanism is used to provide nodes with topological information about their neighbors. Furthermore, there are two system-wide parameters, which are assumed to be known by all the nodes. Maximum delay t_{max} indicates the maximum delay a packet can experience per hop, and a maximum transmission radius r. The network is modeled with the unit disk graph where nodes may communicate directly if their distance is smaller than the fixed r. As a consequence, all links are bidirectional and antennas are omni directional. The node density is expressed in terms of average number of neighbors per node, d.

We have simulated the algorithms presented in this thesis in the connected networks assuming an ideal MAC layer without collisions, and unit disc graph model with possible void areas. We assume that the nodes are placed in the Euclidian plane.

1.6 Organization of the Thesis

The thesis is organized as follows. Chapter 2 gives a review of literature on beaconless geographic routing with guaranteed delivery in wireless sensor networks. Chapter 3 describes the Beaconless Forwarder Planarization algorithm. Chapter 4 describes proximity graphs and beaconless graph construction that will be used in this thesis. Chapter 5 proposes Circlunar Neighborhood graph. Chapter 6 describes the angular relaying algorithm. In Chapter 7, simulation results are presented followed by conclusion in Chapter 8.

Chapter 2 Literature Review

2.1 Geographic Routing

2.1.1 Greedy Forwarding Strategies

In a localized geographic routing scheme, forwarder *S*, currently holding the message, is aware only about the position of its neighbours within the transmission radius and destination *D* (Fig. 2.1).

Various localized protocols have been proposed. Takagi and Kleinrock [32] proposed the first position-based routing scheme, based on the notion of progress. Given a transmitting node *S*, the progress of a node A is defined as the projection onto the line connecting *S* and *D*. In the *Most Forward within Radius* (MFR) scheme [32], the packet is forwarded to a neighbour whose progress is maximal (e.g., node *B* in Fig. 2.1). Nelson and Kleinrock also discussed a random progress method (choosing at random one of the nodes with progress, and adjusting the transmission radius to reach that node), arguing that there is a trade-off between progress and transmission success. Hou and Li discussed the *Nearest Forward Progress* (NFP) method (selecting node *N* in Fig. 2.1).

Finn [10] proposed the greedy routing scheme based on geographic distance. *S* selects neighbouring node *A* (Fig. 2.1) that is closest to the destination among its neighbours. Only neighbours closer to the destination than *S* are considered. Otherwise, there is a lack of advance, and the method fails. A variant of this method is called the *Geographic Distance Routing* (GEDIR) scheme [30]. In this variant, all neighbours are considered, and the message is dropped if the best choice for a current node is to return the message to the node the message came from (stoppage criterion indicating lack of advance). The *Nearest Closer* (NC) method was proposed in [31] (node *N* in Fig. 2.1). In the *compass routing* method (also referred to as the DIR method) proposed by Kranakis, Singh, and Urrutia (e.g., [30]), the message is forwarded to neighbour *C* (Fig. 2.1), such that direction *SC* is closest to direction *SD* (i.e., the angle $\angle CSD$ is minimized).

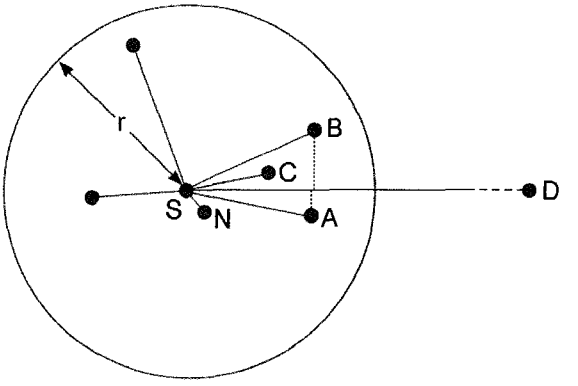


Fig. 2.1. *S* selects *A* in Greedy, *B* in MFR, *C* in compass routing (DIR), *N* in NFP.

The MFR and greedy/GEDIR methods, in most cases, provide the same path to the destination and are loop-free [30]. The hop count for the DIR method is somewhat higher than for the greedy scheme, while the success rate is similar.

All methods have high delivery rates for dense communication networks, and low delivery rates for sparse networks (about half the messages at average degrees below 4 are not delivered) [30]. When successful, hop counts of greedy and MFR methods nearly match the performance of the shortest path algorithm. The DIR method, and any other method that includes forwarding the message to a neighbour with closest direction, are not loop-free (see [30] for counterexample and references).

2.1.2 Recovery Strategies

All greedy routing approaches work well in dense networks, where there is always a neighbor with advance to destination. If this is not the case and the greedy algorithm faces a local minimum, delivery can only be guaranteed, if a recovery from that situation is possible. Recovery strategies have been developed for geographic routing algorithms (see [12], [7] for a survey) and many of them are based on face traversals using a planar subgraph.

The basic idea of planar graph routing is to forward a message along the interiors of a sequence of adjacent faces which are providing progress towards the destination node. Exploration of a single face can be done in a localized way by applying the well known left or right hand rule. Message forwarding according to the left hand rule is similar to sending the message along the edge which is lying next in counter-clockwise direction

from the previous visited edge. The right hand rule in contrast sends the message to the edge lying next in clockwise direction.

When used as a recovery mechanism for a greedy routing failure, planar graph routing may return to greedy routing whenever it encounters a node – this may either be the current message receiver or one of its neighbours – whose distance to the destination is smaller than the distance between the destination and the greedy failure node.

2.1.3 Planar Subgraph Constructions

In general, an arbitrary wireless sensor network graph is not planar. A non-planar graph contains crossing edges which may cause a routing loop during recovery. The planar subgraph is necessary for the recovery strategy to be loop free. Thus, before recovery from greedy routing failure can take place, a planar graph construction mechanism has to be applied in advance.

Prominent subgraph constructions are the Gabriel graph (GG) [14] and the Relative Neighborhood Graph (RNG) [20], but also localized variants of the Delaunay triangulation have been proposed [15], [24], [26].

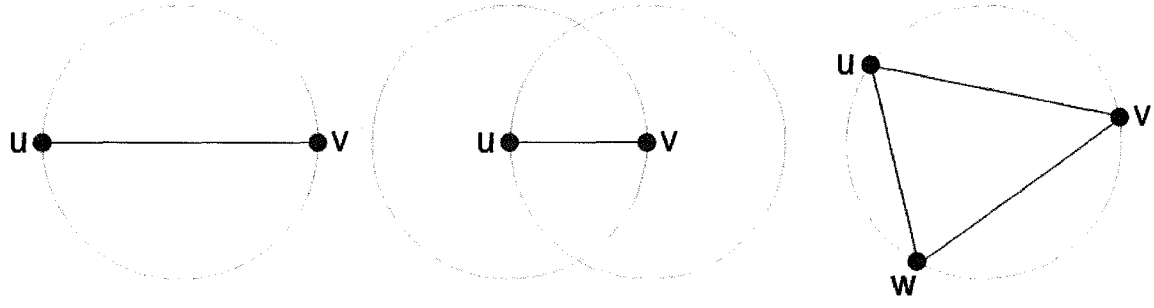


Fig. 2.2. Planar graph construction based on (a) Gabriel graph, (b) relative neighborhood graph, and (c) Delaunay triangulation.

Gabriel graph (GG) [14] – A node u preserves all outgoing edges (u,v) which satisfy that the circle $C(u,v)$ with diameter $|uv|$ contains no other neighbor node than v (see Fig. 2.2(a)).

Relative neighborhood graph (RNG) [20] – A node u preserves all outgoing edges (u,v) which satisfy that the intersection of the circles with center u, center v, and radii $|uv|$ contains no other node than v (see Fig. 2.2(b)).

Localized Delaunay triangulation (LDT) [15], [24], [26] – Each node computes the Delaunay triangulation on its own neighbor set. The Delaunay triangulation in general contains all triangles which satisfy that the circle passing through the triangle end points does not contain any other node (see Fig. 2.2(c)). From the subset of outgoing Delaunay edges each node preserves all outgoing edges which are preserved by the node on the other edge end point as well.

Notice that, unlike Gabriel graph and RNG, Delaunay triangulation of a set of nodes cannot be constructed in a localized manner, i.e. it requires 2-hop information. This construct is given only as an example of planar graph construct. In this thesis we use the planar constructs that require 1-hop neighborhood information only.

2.2 Beaconless Routing

2.2.1 Beaconless Forwarding

Conventional geographic routing strategies generally make three major assumptions. Usually the sender node is assumed to be aware of the geographic position of itself, the destination node position, as well as the position of all its 1-hop neighbors. The first assumption is fulfilled by the cheap GPS-receivers available at each sensor node. The second assumption is achieved by applying one of the existing location services [28]. Finally, the last assumption is granted by the periodic beaconing that each active sensor node must participate in.

The main advantage of any beaconless routing scheme is its operation in the absence of knowledge about the immediate neighborhood, i.e. the absence of the periodic beacon messages. In beaconless protocols the forwarder node can make a next hop routing decision without periodical advertisement of each neighboring nodes' positions using small beacon messages. The forwarder doesn't need to know anything about the existence or positions of its neighbors. One obvious advantage of this approach is the use of the actual current positions of all neighbouring nodes instead of the last known cached values. This increases routing accuracy and robustness against the topological changes in the communication network. The other advantage is the absence of the additional load which the beacon messages impose on the network traffic.

The second distinctive property of the beaconless routing schemes is the implicit neighbor selection by contention. Each neighbor is aware of a specific delay function and when it receives a packet, either control or data, it can start a timer. When the timer expires the neighbor can make a deterministic decision on whether to respond with another control packet, or rebroadcast the data packet, or remain quiet. The details differ depending on the specific protocols.

The timer delay function controls the next hop and its selection may be based on different criteria, like distance to the destination, remaining energy, load, previous usage, fault rate, random values, or some combination of these. The basic criteria for the timer function are: 1) it should select a good forwarder (e. g., based on the progress), and 2) it should differentiate the length of times at the different nodes in order to avoid simultaneous or almost simultaneous selection.

The concept of a forwarding area is another important notion used in beaconless geographic routing. A forwarding area consists of a set of sensor nodes that can mutually communicate with each other. Forwarding areas are described using geographical constraints, such that a node can determine whether it belongs to a forwarding area or not using only available location information. Using only the nodes in the forwarding area the beaconless routing scheme can avoid unnecessary packet acknowledgements; it can also limit the set of possible next-hop candidates to neighbours with positive advance to destination.

There are three reasonable possibilities for the selection of the forwarding area: a 60° sector which covers the largest area of all; a circle with diameter r (transmission range) which contains more nodes that make big progress toward the target; and a Reuleaux triangle [13] with a width of r which contain fewer nodes close to the forwarder (see Fig. 2.3)

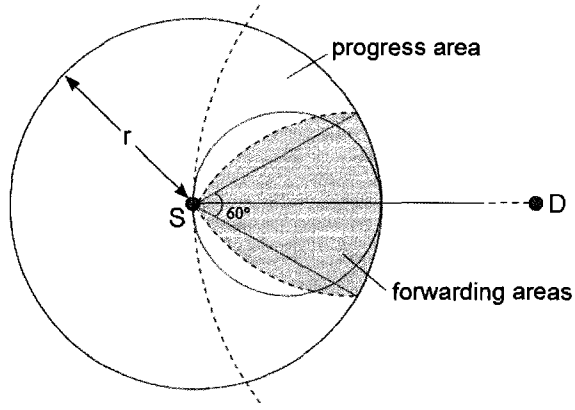


Fig. 2.3. Forwarding areas: 60° sector, Reuleaux triangle (shaded), and circle.

Assuming that each node in the network is aware of the specific forwarding area description and the specific timer delay function, the basic steps of the beaconless routing algorithm can be described as follows [33]:

1. Node *S* broadcasts the packet.
2. All neighbouring nodes receive the packet. Nodes outside the forwarding area discard the packet right away.
3. All nodes in the forwarding area start a timer, if the destination has not been reached.
4. The first node where the timer expires continues with 1.

A specific algorithm must define which forwarding area is used, how to either guarantee that the forwarding area is not empty or apply a recovery strategy in case of failure, and the duration of the timers.

2.2.1.1 Beacon-Less Routing

Beacon-Less Routing (BLR) [17] algorithm starts in Basic mode, in which the packets are forwarded using one of the greedy routing strategies, e.g. Greedy or MFR. The forwarding area is a 60° sector from the previous node's position towards the destination location with the radius equal to transmission range *r*. Any node in this forwarding area calculates a delay based on it's progress to destination. In case BLR is based on greedy routing strategy the delay is the decreasing function of the node's progress. Thus the node with the smallest delay retransmits the packet first. No acknowledgement message is necessary since all nodes in the forwarding sector can overhear this retransmission and cancel their timers. When the basic mode fails (i.e. the

forwarding sector is empty), the BLR algorithm switches to backup mode which provides a fallback mechanism to recover from greedy failure. As soon as the packet arrives at a node closer to the destination than where it entered the backup mode, it switches back to the basic mode again.

In request-response backup mode approach the node broadcasts the request message and all neighboring nodes respond. This approach is similar to beaconing. In clockwise-relaying backup mode approach all neighboring nodes introduce a second timer with delay based on angle between the node itself, the previous node and the destination node. With this delay function any node with forward progress relays the packet before any node with backward progress in a clockwise order. After the best candidate retransmits the packet, the forwarder has to transmit the STN-packet (Successful Transmission Notification) to let other nodes cancel their timers. This approach avoids any beaconing mechanism; however it does not guarantee delivery.

2.2.1.2 Contention-Based Forwarding

Contention-Based Forwarding (CBF) [13] works in three steps: first, the forwarding node transmits the packet to all neighbors. Second, the neighbors compete with each other for the right to forward the packet. During this contention period a node determines how well it is suited as a next hop for the packet. To do this, each node starts the timer with the delay calculated based on the progress represented by the difference between the distances from forwarder to destination and from node itself to destination. The node with the most progress has the timer with the smallest delay. Finally, the node that wins the contention suppresses the other nodes and becomes the next forwarding node. On the second step the suppression is needed to avoid packet duplications. Several suppression schemes are presented. In the basic suppression scheme the nodes which overhear the packet transmission by the contention winner cancel their timers and remain quiet. In the area-based suppression scheme further restrictions are applied to the location of the nodes which are allowed to participate in the contention period. One variant is to only allow the nodes within the Reuleaux triangle [13] in the direction towards destination node (see Fig. 2.4). Another variant is to consider the possible candidates within the circle with diameter r placed towards the destination (Fig. 2.4). The first variant is preferred since the Reuleaux triangle much better covers the area with good forwarding progress. Finally,

the active selection of the next hop prevents all forms of packet duplications at the cost of additional control messages. This scheme works as follows: the forwarding node broadcasts an RTF control packet (Request To Forward) instead of immediately broadcasting the packet. All candidate nodes start a progress-based contention timer. When the timer expires, the candidate neighbor sends back the CTF packet (Clear To Forward). After receiving one (or more) CTF packets the forwarder selects the candidate which offers the largest progress towards destination and transmits the packet to this node. Thus the forwarding node acts as a central authority deciding which node is selected as the next hop.

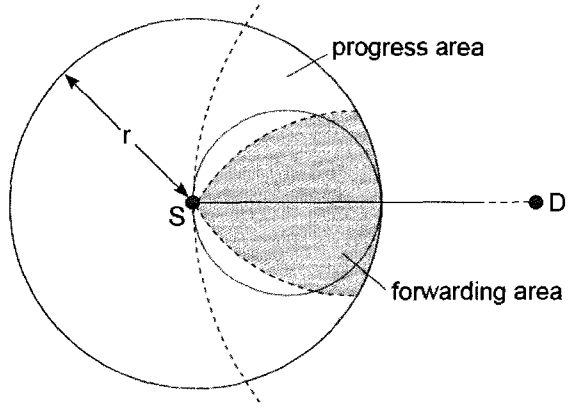


Fig. 2.4. Forwarding areas in CBF. Reuleaux triangle is shaded.

The CBF algorithm does not include the strategy for recovery from the greedy failure and thus cannot guarantee packet delivery.

2.2.1.3 Implicit Geographic Forwarding

Implicit Geographic Forwarding (IGF) [1] is a combined routing/MAC protocol that utilizes beaconless greedy forwarding strategy. Forwarder S starts by sending an ORTS packet (open RTS). All potential candidates for packet transmission are located within the 60° sector in the direction towards destination. This forwarding area ensures that, first, a message is propagated on a progressive path towards destination; and second, that every node within this area is capable of hearing one another, to prevent interference between candidate nodes. The assumption is that the network density is high enough for the forwarding area to be non-empty. Candidates set progress-based *CTS_Response* timer, then the node with the shortest timeout responds with CTS. The timer delay is defined as a decreasing function of two parameters: increased distance toward the

destination and energy remaining. Other candidates cancel their timers to avoid duplicate CTSs. The forwarder sends the packet to the first CTS sender, which is followed by ACK message if the packet is received successfully. In case of empty forwarding area the forwarding area shift technique is mentioned as the means to recover from local optima. However, the details of this technique are omitted, and this technique will not prevent possible packet loss, and thus IGF cannot guarantee delivery.

2.2.1.4 Blind Geographic Routing

In Blind Geographic Routing (BGR) [33] the forwarding area is an implementation-dependent choice. When the packet is transmitted the forwarder starts the timer with maximum delay and waits for neighbour replies. Each neighbour in the forwarding area starts a contention timer; the timer of the node with the largest progress expires first, and that candidate becomes the next forwarder by re-transmitting the packet. If no node forwards the packet, the forwarding area is assumed to be empty and up to two different retries are performed with forwarding area turned 60° to the left or right. These attempts constitute the recovery strategy and if they also fail, the packet is dropped. Thus the delivery is not guaranteed. During recovery nodes that are further from the destination than the forwarder may reply; the maximum timer delay value is adjusted to include these replies into consideration.

The main innovation of the BGR is a technique to solve the problem of two nodes forwarding the packet simultaneously. When this happens, the neighbours of the first node start their contention timers, but cancel them right away since they receive the same packet from the second node. The second node is deemed the contention winner and thus all timers, including recovery timers for both nodes, are cancelled and the packet is never re-transmitted. This situation applies when the forwarding areas of both nodes contain the same set of neighbours. To solve this problem, the current number of hops is included in the packet header. Before cancelling the contention timer every candidate node compares the hop count of the received packet to that of the stored packet for which it contends. If they are equal, the timer is not cancelled.

2.2.1.5 Geographic Random Forwarding

Geographic Random Forwarding (GeRaF) [35] is based on the assumption that sensor nodes have a means to determine their location, and that the positions of the final destination and of the transmitting node are included in each message. Data packets are routed by selecting the relay node which is most favorably located towards the destination. This selection is made based on the relative location of the transmitter, relay and destination.

The protocol works as follows. When a node has a packet to send, it listens to the radio channel for certain time duration T_{sens}. If some activity is detected, the node backs off and schedules a reattempt at a later time. If the channel is sensed idle during this entire interval, the node starts its transmission by broadcasting an RTS message, which contains the location of the destination as well as its own. After sending the RTS, the node listens for CTS message from potential candidates. If only one CTS message is received, it starts the transmission of the data packet. If no CTS are heard, it will send the CONTINUE message and listen for CTSs again, timing out after N_p empty CTS slots ("empty cycle"). If a collision took place it will send a COLLISION message which triggers a collision resolution algorithm. Each neighbor node which receives the initial RTS from the current transmitter will determine its own priority as a relay. This priority is based on subdividing the relay region (progress area) into N_p regions $A_1,...,A_{N_p}$ such that all points in A_i are closer to the destination than all points in A_j for $i, j = 1,...,N_p - 1$. Possible choices of these regions may be to take all with the same area or to quantize the advancement in N_p equal levels (see Fig. 2.5). All neighbor nodes start a contention among each other. If the candidate node in A_i hears valid CTS from another neighbor with higher priority, it cancels its own CTS and remains quiet. If the node hears nothing or hears only the CONTINUE messages for the duration of $i-1$ CTS slots, it sends its own CTS message. The winner of the contention will receive the data packet from the transmitter. If there was a collision in case two nodes in the same region A_i send CTS at the same time, a binary splitting collision resolution algorithm is executed. In this case, all nodes involved will decide with probability $1/2$ whether or not to send again in

the next CTS slot. If nobody sends, this random decision is repeated in the next slot. Otherwise only those who have sent will survive, until there is a single survivor.

This approach works well in the networks with sufficient density to guarantee that the forwarding area consisting of all regions $A_1,...,A_{N_p}$ is not empty and the empty cycle leading to packet drops never occur. Thus the GeRaF algorithm does not guarantee data packet delivery.

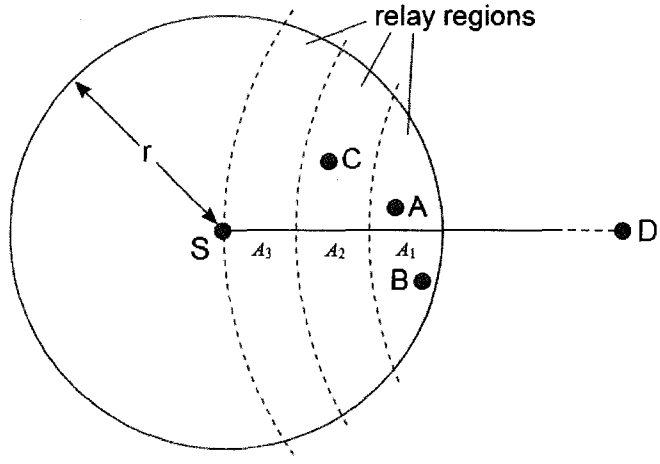


Fig. 2.5. GeRaF relay regions variant. Nodes A and B compete, node B wins, node C drops out.

2.2.1.6 Priority-based Stateless Geo-Routing

Priority-based Stateless Geo-Routing (PSGR) [34] protocol exploits two important concepts: autonomous prioritized acknowledgements and dynamic forwarding zone formation based on the sensor node density estimated on the fly. It also addresses the communication void problem by introducing two recovery approaches: rebroadcast and bypass.

The basic idea of prioritized acknowledgement is to assign acknowledgement precedences (AckP) to all candidate nodes such that they can respond to a forwarding request without contention among each other. The PSGR protocol chooses to assign the same AckP value to all nodes positioned within the same forwarding zone. The forwarding zones are formed on the fly by dividing the whole progress area into a number of sub-regions. Two variants of such subdivision are suggested (see Fig. 2.6). According to the first variant the progress area is partitioned into Z forwarding zones based on the distance to destination (Fig. 2.6(a)). In the second variant the progress area is first split into three "sectors" with the help of Reuleaux triangle towards the

destination; then each sector is further sub-divided based on distance to destination (Fig. 2.6(b)). The number Z in both variants is dynamically derived from the estimated node density on each step. The density estimation technique requires that each node maintains the record of the number of unique nodes residing in its vicinity within certain time window. This is obtained from the messages the node overhears during this time window. This approach works well in assumption of dense network traffic and fails to estimate the correct density otherwise. Although PSGR attempts to form zones that have only one candidate neighbor in order to avoid collisions, there is no guarantee the there is only one candidate in each forwarding zone.

The PSGR protocol works as follows. After receiving the "forwarding probe" packet from the current forwarder, each neighbor node sets a timer based on it's location within a certain forwarding zone. After the timer expires, and provided that no other candidate has acknowledged the probe yet, the candidate node sends an acknowledgement packet. The packet holder then forwards the data packet to the first acknowledger, which becomes the next forwarder. The latter transmission is overheard by all neighbors and forces them to stop their timers. This process is repeated until the delivery succeeds or fails due to the empty forwarding area.

Two methods are suggested for solving the communication void problem when the forwarding area is empty: rebroadcast and bypass. The first one is based on the belief that a candidate may exist near the void forwarding area. Thus, after the first failure to receive acknowledgement, the forwarder waits for a certain period of time and then broadcasts the same forwarding probe again, this time with the maximum transmission range possible. The process repeats until the acknowledgement is received or the maximum allowed number of rebroadcasts is exceeded. The rebroadcast approach doesn't work at all in case a permanent void region is encountered.

The bypass recovery adopts the right-hand rule face routing idea and works as follows. During the regular forwarding process, the sensor nodes located in the current forwarder's transmission range but outside the progress area anticipate the potential bypass events by setting their bypassing acknowledgement timers when receiving the initial forwarding probe packet. The timer delay is selected so that it is longer than the delay of any candidate located within the forwarding area and at the same time so that the

duplicate messages from potential bypass nodes are suppressed. The potential bypass node cancels its timer once it overhears a message from the holder, any candidate within the forwarding area or other bypass node. Otherwise it acknowledges the packet holder. At this point the forwarding switches from forwarding (greedy) to bypassing (recovery) mode. The bypassing switches back to forwarding as soon as possible, i.e. as soon as the node closer to destination as the local minimum is reached. The current bypassing node broadcasts the bypassing probe, and all its neighbors start the bypassing timer. To prevent loops, bypassing nodes keep the track of the packets they have previously received for bypass and exclude themselves from being a bypass candidate when they receive the same bypassing probe again. Thus the PSGR with bypass recovery is loop free only in the sense that the packet is dropped if the loop is about to be formed. There is no formal planar graph structure used during recovery. Therefore, neither the PSGR with rebroadcast recovery nor PSGR with bypass recovery can guarantee packet delivery.

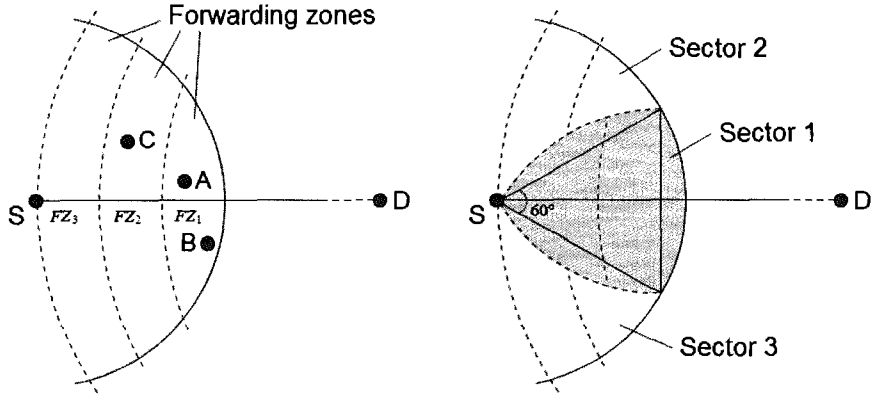


Fig. 2.6. PSGR forwarding zones are formed based on distance to destination.

2.2.2 Beaconless Recovery Algorithms

All approaches described in the previous section 2.2.1 work well in dense networks, where there is always a neighbor closer to the destination. If this is not the case and the greedy algorithm faces a local minimum, delivery can only be guaranteed, if a recovery from that situation is possible.

While the recovery problem is well studied for geographic routing algorithms, the beaconless strategies leave room for improvement. In beaconless routing, the term "recovery" often refers to the strategies which enlarge the set of possible candidates, if

the forwarding area is empty, but do not guarantee delivery. CBF, IGF and BGR use this kind of recovery strategy. BLR includes the backup mode which either falls back to using beacon message for obtaining the full neighborhood knowledge prior to making the next hop decision or doesn't guarantee delivery due to possible loops. PSGR [34] contains a more sophisticated recovery mechanism; however the delivery is questionable, as no crossing-free subgraph is considered.

2.2.2.1 BLR Backup mode

In BLR Backup mode [18] (also called Request-response approach in [17]) the forwarder broadcasts a request and all neighboring nodes respond. If a node is closer to the destination, it becomes the next hop. Otherwise the forwarder constructs a local planar (Gabriel) subgraph from the position information of the neighbors and forwards the packet using the right-hand rule. The position when entering backup mode is stored in the packet. Greedy forwarding is resumed as soon as the node closer to the destination is encountered.

Request-Response can be regarded as reactive beaconing, because all neighbors are involved in exchanging position information. The following protocols use an approach, that we classify as Select and Protest: they determine possible neighbors of a planar subgraph by a contention process and allow protests afterwards to correct wrong decisions.

2.2.2.2 No Beacon FACE Algorithm

No Beacon FACE (NB-Face) [27] algorithm is a beaconless variant of the face routing algorithm. The delay function depends on the angle between candidate, forwarder and previous hop such that the first candidate in (counter-) clockwise order responds first. If this node is not a neighbor in the Gabriel graph, then other nodes may protest.

The forwarder S broadcasts the *Rreq* control packet with location information of itself and also the previous node, from which it got the packet in the first place. Each candidate node calculates the angle θ between the previous node, the forwarder, and itself using the location information available. Then each candidate neighbor sets the times based on that angle. To determine the minimum angle in (counter-) clockwise direction, a monotonically increasing (decreasing) function of θ can be used to calculate the delay.

The timer of the node with the minimum angle in (counter-) clockwise expires first, and that node broadcasts the proposal message *Prop*. After this the protests can be sent by the nodes which overhear the *Prop* message and are located inside the GG circle of forwarder and the candidate node. Such nodes would broadcast the *Nack* (Negative Acknowledgement) message. The situation of cascading protest is not mentioned. The forwarder waits for the response messages for a certain period of time and then sends the finish message *Fin*. The packet is sent to the last known *Prop* or *Nack* sender.

The NB-FACE algorithm is similar to a variant of our Angular Relaying scheme (Chapter 6). However, we will see that NB-FACE doesn't always yield optimal results.

2.2.2.3 Guaranteed Delivery Beaconless Forwarding

Guaranteed Delivery Beaconless Forwarding (GDBF) [5], [6] is a generic framework for beaconless routing that can be applied to location based schemes like GFG and thus can guarantee delivery if the underlying protocol is a guaranteed delivery protocol. The GDBF runs in two modes: greedy and recovery, and works as follows. First, the current packet holder broadcasts the Ready To Send (RTS) message to all its neighbors. The packet contains the request to send the message and a bit indicating which mode – greedy or recovery – is currently in progress. Second, all neighbors compete with each other for the right to be the next hop. During this step each neighbor sets the position-based timer with the different delay for greedy and recovery mode. If the neighbor node overhears any messages from other neighbors while waiting it may stop its timer and cancel the response. The response, if sent, is in the form of Clear To Send (CTS) message. Finally, the current sender node decides which neighbor is the most suitable and forwards the data packet to that neighbor. The details differ in greedy and recovery modes.

In greedy mode the timer delay is an increasing function of the distance to destination: the neighbor with the smallest distance will have the shortest timeout. The timer is set only by the nodes in the progress area, i.e. by those which are closer to the destination than the current sender. As soon as the first CTS is received the sender forwards the packet to the CTS sender, thus forcing all other neighbors to stop their timers. The operation of GDBF in greedy mode is thus similar to the GeRaF algorithm [35] (see also section 2.2.1.5)

In recovery mode the timer delay is the increasing function of the distance to current sender S rather than the destination. Nodes closer to the sender have shorter timeouts. Once the timer expires, neighbor A responds with a CTS to S. If any other neighbor X can hear this CTS, it cancels its timer if it finds node A in the Gabriel circle over $|SX|$. The number of CTSs received by the current sender in recovery mode is always greater or equal than the number of actual Gabriel graph neighbors. Having the list of all CTS senders, the sender follows the underlying protocol (e.g. GFG) and sends the packet to the best suited next hop neighbor. Since the selected neighbor may not be the actual planar graph neighbor, the sender then waits for "stop" messages (protests) from the potential "witnesses". S then selects a different neighbor until no "stop" message is received. The underlying routing protocol regulates when the recovery is reached and thus when GDBF should switch back to greedy mode.

The authors of GDBF claim that in simulations they didn't encounter the case when the node which is not an actual GG neighbor replied with CTS in recovery mode. In this thesis we provide examples of such a case and, moreover, show that the "stop" messages (protests) are unavoidable during recovery. The Beaconless Forwarder Planarization (BFP) scheme presented in this thesis is a generalization of the recovery mode operation in the GDBF framework.

2.2.3 Beaconless Routing: Summary

The following table summarizes the main characteristics of the beaconless routing algorithms described above.

Protocol	Empty Forwarding Area	Recovery strategy	Delivery guarantee
BLR	use MFR area	Beaconing + face routing	yes
CBF	use entire progress area	Left open	?
IGF	rotate forwarding area	–	no
BGR	rotate forwarding area	–	no
GeRaF	– *	–	no
PSGR	– *	Rebroadcast or Bypass	no
NB-FACE	– **	Clockwise timeout and Gabriel neighbor selection	yes
GDBF	– **	Distance-based timeout, Gabriel neighbor selection	yes

*) Forwarding area covers the complete progress area
**) Forwarding area covers the complete transmission area

Table 2.1. Beaconless routing protocols and their recovery methods

Chapter 3 Beaconless Forwarder Planarization

The basic problem of beaconless protocols is that they cannot rely on 1-hop knowledge. But this knowledge is necessary to build a planar subgraph. Thus, in a recovery situation, the forwarder has to gather information and this is connected with the exchange of messages. In contrast to the Request-Response approach of BLR [17], where all neighbors announce their positions upon request, we follow the idea of GDBF [6] to reduce the message overhead.

Beaconless Forwarder Planarization (BFP) is a general scheme, which can be used to construct different proximity graphs, such as Gabriel graph or RNG. The BFP algorithm is described in the following. Its message complexity depends on the chosen subgraph. We will later discuss appropriate subgraph constructions and analyze the message complexity.

In the following we describe the BFP algorithm for the network layer assuming an ideal MAC layer without collisions. Although we adapt the request to send (RTS) and clear to send (CTS) names for the messages sent in BFP, these don't refer to the MAC layer messages, but rather the control messages required for the correct operation of the BFP algorithm.

3.1 The Beaconless Forwarder Planarization Algorithm

The Beaconless Forwarder Planarization (BFP) algorithm consists of two phases, the selection and the protest phase. $N(u,v)$ denotes the proximity region of the chosen subgraph, e.g. the Gabriel circle or the RNG lune, over (u,v) (see Fig. 2.2). Here, the proximity region of two nodes u and v is a portion of the plane that contains points relatively close to both of them, and that does not contain any graph nodes other than u and v.

3.1.1 Selection Phase

The forwarder u broadcasts a request to send (RTS) message (including its own position) and sets its timer to t_{max}. Each candidate v sets its contention timer, using the following delay function:

$$t(d)=\frac{d}{r}t_{max} \tag{1}$$

where $d = |uv|$ is the distance to forwarder, r is the transmission radius, and t_{max} is the maximum timeout. When the contention timer expires, a candidate answers with a clear to send (CTS) message. If a candidate v receives the CTS of another node x which lies in the proximity region $N(u,v)$, then v cancels its timer and remains quiet. We call this mechanism *suppression* and the candidate being suppressed a *hidden node*. Hidden nodes listen to other nodes after their timer expired. If a hidden node x receives the CTS of another node w with $x \in N(u,w)$ (as shown in Fig. 3.1), then w violates the proximity condition and x adds w to the set of violating nodes $S(x)$. We call (u,w) a violating edge. See also Fig. 3.2.

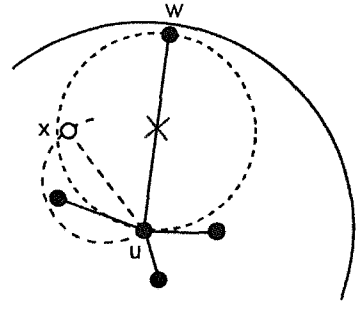


Fig. 3.1. *(u,w)* is a violating edge because of hidden node x.

3.1.2 Protest phase

In the second phase, the hidden nodes protest against violating edges. If the set of violating nodes $S(x)$ is not empty, the hidden node x starts its timer, using the same delay function as in the first phase (closest candidates protest first). If x overhears a protest from another hidden node y, then the set of violating nodes has to be checked: A node v can be removed from $S(x)$, if $y \in N(u,v)$. In other words, node v can be removed from $S(x)$ if some other hidden node y has already protested against it. When

the timer expires and $S(x)$ is not empty, x sends the protest message. The forwarder removes violating edges when it receives protests and finally obtains a planar subgraph.

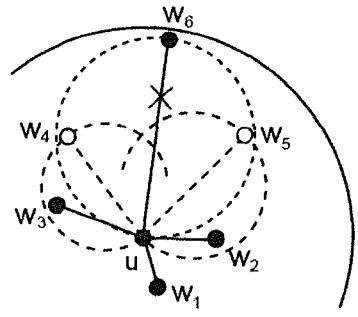


Fig. 3.2. BFP: Nodes respond in the order w_1, w_2, w_3, w_6; w_4 and w_5 are hidden. w_4 protests against w_6. w_5 cancels its protest against w_6.

3.2 *BFP: Algorithm Pseudo Code*

Suppose $G=(V,E)$ is the unit disk graph and $N_{pl}(u)$ is the planar, i.e. either GG, CNG (see Chapter 5 below) or RNG neighbourhood of node u. The task of BFP is to get $N_{pl}(u)$ for all nodes $u \in V$.

Algorithm 1. Beaconless Forwarder Planarization, forwarder node

```
1:   for each node u in V
2:       u sends RTS
3:       u sets timer T1 to tmax
4:       for each CTS received by u
5:           u saves CTS to Npl(u)
6:       end for
7:       after expiry of timer T1 u sets timer T2 to tmax
8:       for each Protest from node x received by u
9:           u removes all violating nodes v from Npl(u) *
10:      end for
11:  end for
```

*) node v is considered violating node if $x \in N(u,v)$, where $N(u,v)$ is a proximity region of GG, CNG or RNG over (u,v).

Now suppose $N(u)$ is the neighborhood of node u and $C(v)$ is the list of CTSs overheard by node v.

Algorithm 2. Beaconless Forwarder Planarization, neighbor node

```
1:   for each node v in N(u)
2:       v sets timer T1 to t(d) according to equation 1
3:       for each CTS overheard by v
```

4: v saves CTS to $C(v)$
5: **end for**
6: after expiry of timer T_1
7: **if** $C(v)$ is empty **then**
8: v sends CTS
9: **else**
10: v sets timer T_2 to $t(d)$ according to equation 1
11: **for** each Protest from node x received by v
12: v removes all violating nodes w from $C(v)$ *
13: **end for**
14: after expiry of timer T_2
15: **if** $C(v)$ is not empty **then**
16: v sends Protest to u with its own position
17: **end if**
18: **end if**
19: **end for**

*) node w is considered violating node if $x \in N(u,w)$, where $N(u,w)$ is a proximity region of GG, CNG or RNG over (u,w).

3.3 *Face Routing with BFP: Algorithm Pseudo Code*

Suppose $G=(V,E)$ is the unit disk graph and $N_{pl}(u)$ is the planar neighbourhood of node u. The task of face routing with BFP is to deliver the packet from the node where greedy routing failed to either the recovery or destination node.

Algorithm 3. Face routing with BFP, forwarder node

1: Let u be the current forwarder or the packet
2: Let m be the local minimum distance to destination
3: **repeat**
4: u sends RTS
5: u sets timer T_1 to t_{max}
6: **for** each CTS received by u
7: u saves CTS to $N_{pl}(u)$
8: **end for**
9: after expiry of timer T_1 u sets timer T_2 to t_{max}
10: **for** each Protest from node x received by u
11: u removes all violating nodes v from $N_{pl}(u)$ *
12: **end for**
13: u sorts all nodes in $N_{pl}(u)$ by angle, smallest first
14: u sends the packet to $N_{pl}(u)_1$, i.e. $u = N_{pl}(u)_1$ **
15: **until** packet delivered to destination or u is closer to destination than m

*) node v is considered violating node if $x \in N(u,v)$, where $N(u,v)$ is a proximity region of GG, CNG or RNG over (u,v).

**) assuming that the previous packet sender is not part of $N_{pl}(u)$.

3.4 *Duplicate Protests Against The Same Violating Node In BFP*

The maximum distance between any two nodes in the proximity region of Gabriel graph is always less than or equal to the transmission radius r. In other words, any two nodes in the Gabriel graph proximity region over (u,v) can hear each other. This is, however, not true for proximity regions of RNG and CNG. Thus, in the latter two graphs the duplicate protests against the same violating node are unavoidable.

Our simulations show that duplicate protests in the CNG almost never happen, whereas they happen quite often in the RNG. Consider the following example (RNG edges are marked as red solid, violating edges are red dashed).

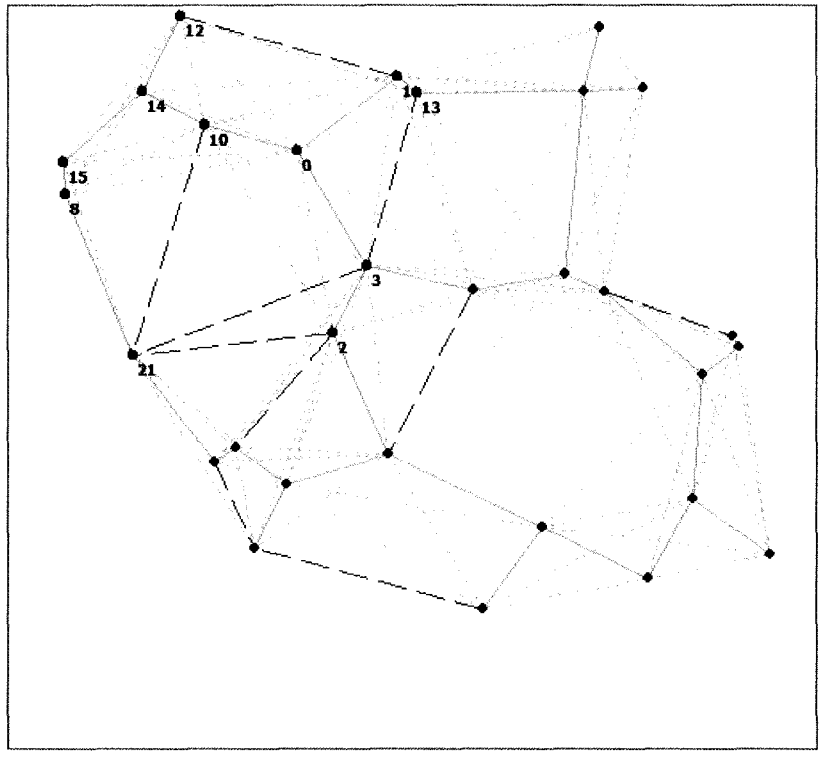


Fig. 3.3. Example of duplicate protests against the same violating node in BFP on RNG

The task of BFP is to construct RNG neighbourhood of every node without beaconing. Consider node 10. The list of neighbours sorted by timeouts (equation 1) is given in the table below:

Node Id	14	0	12	15	8	1	13	3	2	21
Timeout	2.77	3.62	4.56	5.55	5.98	7.53	8.07	8.37	9.83	9.87

The BFP algorithm runs as follows in node 10.

Selection Phase:

- 14, 0 send CTS;
- 12, 15, 8 cancel CTSs because of 14;
- 1, 13, 3, 2 cancel CTSs because of 0;
- 21 sends CTS.

Protest Phase:

- 12 has no one to protest against;
- 15 sends PROTEST; Edge (10,21) eliminated because of 15;
- 8 heard protest from S 15 and has no one to protest against;
- 1, 13 have no one to protest against;
- 3 didn't hear protest from 15 and sends duplicate PROTEST against 21;
- 2 heard protest from 3 and has no one to protest against.

Thus in RNG and CNG the number of protests can be either less (when multiple edges are eliminated as a result of single protest, see section 3.5), equal or greater (when there are duplicate protests against the same violating node) than the number of violating edges.

3.5 BFP: Single Protest Eliminating Multiple Violating Nodes

Consider the following network. (GG edges are marked as light-blue solid, violating edges are blue dashed).

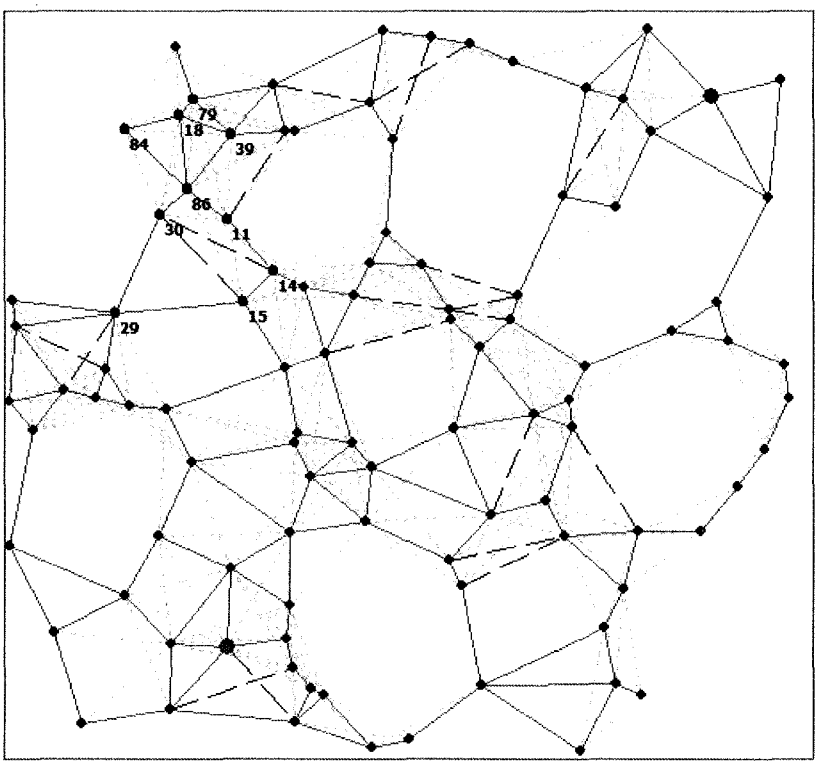


Fig. 3.4. Example of single protest eliminating multiple violating nodes for BFP on GG

The task of BFP is to construct GG neighbourhood of every node without beaconing. Consider node 30. The list of neighbours sorted by timeouts (equation 1) is given in the table below:

Node Id	86	11	84	18	39	29	15	14	79
Timeout	2.81	4.88	7.11	7.93	8.06	8.26	9.02	9.23	9.31

The BFP algorithm runs as follows in node 30. Neighbour 86 sends CTS; neighbours 11, 84, 18, 39, and 79 overhear CTS from 86 and become hidden; neighbours 29, 15, and 14 send CTSs. Now the protest phase begins. Node 11 is the first one to send the protest. As a result of this protest, node 30 eliminates two violating neighbours at once: nodes 14 and 15. Thus only one protest is enough to construct plain neighbourhood of node 30.

Chapter 4 Proximity Graphs and Beaconless Subgraph Construction

The BFP algorithm can be based on different proximity graph constructions, in order to obtain a planar communication graph (here, it means that the graph is a planar embedding). Most prominent subgraph constructions are Gabriel graph and RNG (see [8]):

Definition 1: The Gabriel graph (GG) of a node set V contains an edge *(u,v)*, iff $|uv|^2 \leq |uw|^2 + |vw|^2$ for all $w \in V, w \neq u, v$.

Definition 2: The relative neighborhood graph (RNG) of a node set V contains an edge *(u,v)*, iff $|uv| \leq \max\{|uw|, |vw|\}$, for all $w \in V, w \neq u, v$.

The definition implies that two nodes u and v are adjacent, if the so-called proximity region over *(u,v)* is empty. The *proximity region* of two nodes u and v is a portion of the plane that contains points relatively close to both of them. The *proximity condition* is said to be met when the proximity region doesn't contain any graph nodes other than u and v. The *proximity graph* is a graph such that (1) for each edge (u,v) the proximity condition is met, and (2) for each pair of non-adjacent vertices u, v the proximity region contains at least one other node.

We denote the proximity region as $N(u,v)$. In case of the Gabriel graph, the proximity region $N_{GG}(u,v)$ is a circle with diameter $|uv|$; in case of the RNG, $N_{RNG}(u,v)$ is a lune over uv, i.e. a lune formed by the intersection of two circles of the same radius $|uv|$ having u and v as the center points (see Fig. 2.2). In this thesis we assume that all distances are different in order to avoid degenerated cases. However, in case of equal distances can be handled by using $|uv| = \left(\|u - v\|_2, key(u), key(v)\right)$ as distance measure [26], where $key(\cdot)$ is based on the node ID. In a similar way, a modified

RNG with a constant maximum node degree can be obtained that is still connected on degenerated node sets [25].

The choice of the subgraph determines the message efficiency of the BFP algorithm. In the following we will identify the crucial properties to construct a planar and connected subgraph with as few messages as possible.

4.1 Basic Requirements

We consider only undirected, planar, and connected proximity graphs. The proximity region of these graphs is symmetric, it contains at least the Gabriel circle, and it is not larger than the RNG lune.

Lemma 1: The RNG lune is the maximum proximity region to preserve connectivity.

Proof: Let u, v, w be nodes of an undirected proximity graph, and let $L(u,v)$ denote the RNG lune over (u,v), i.e. the intersection of two circles with radius $|uv|$ centered at u and v. Suppose the proximity region of (u,v) is larger than $L(u,v)$. Then there is a point w outside $L(u,v)$ (i.e. $|uw| > |uv|$ or $|vw| > |uv|$) that belongs to the proximity region and thus invalidates the edge (u,v). If $|uw| < |vw|$ then $u \in L(v,w)$, which disconnects v. Otherwise, $v \in L(u,w)$, which disconnects u. ■

Lemma 2: The Gabriel circle is the minimum proximity region to obtain planarity.

Proof: Let $C(u,v)$ denote the Gabriel circle over (u,v), i.e. the circle having $|uv|$ as diameter with its interior. Let m be the midpoint of (u,v). Suppose the proximity region is smaller than $C(u,v)$. Then there is a node w inside $C(u,v)$ with $|mw| < |mu|$, while (u,v) is a valid edge. As the graph G is undirected, the proximity region is symmetric; and this implies that there is another point w' which can be constructed by rotating w by 180° around the midpoint m. Then the circle $C(w,w')$ is inside $C(u,v)$ and empty (since $|mw| = |mw| < |mu| = |mv|$). Therefore, (w,w') is a valid edge, and it intersects (u,v) in the midpoint, which is a contradiction.

The graph is planar, if the proximity region contains $C(u,v)$: If $C(u,v)$ is empty, then the empty circle rule of the Delaunay Triangulation is also fulfilled for any three nodes. Thus, G is a subgraph of the Delaunay Triangulation, which is planar. ■

4.2 *Hidden Nodes and Suppression*

The construction of Gabriel graph or RNG is based on the proximity region, which is an empty circle or an empty lune. BFP makes use of this fact to reduce messages: Candidate nodes are suppressed, i.e. they remain quiet, if they would violate this condition.

Definition 3: The suppression region of a node v with respect to u contains all points w with $v \in N(u,w)$, where $N(u,w)$ denotes the proximity region of an edge *(u,w)*.

Fig. 4.1 shows the suppression region for Gabriel graph and RNG. In case of the Gabriel graph, w is suppressed, if $\angle uvw > 90°$, and this implies that the border of the suppression region is orthogonal to *(u,v)*. In case of the RNG, $|vw|<|uw|$, and this means that the perpendicular bisector of *(u,v)* marks the border of the suppression region.

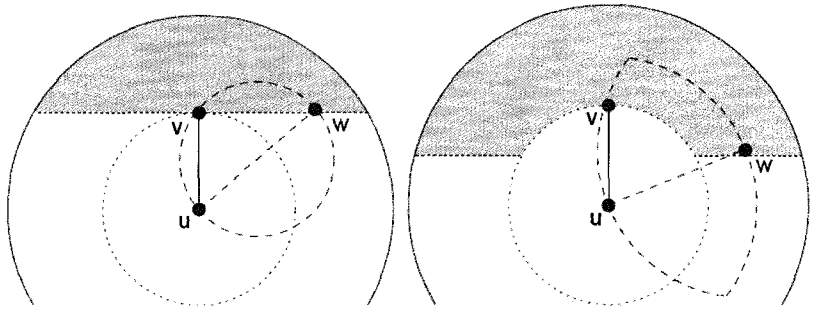


Fig. 4.1. Suppression region for GG and RNG: A node w in the shaded area is not a valid neighbor of u, because v would be inside the Gabriel circle or the RNG lune.

4.3 *Ordered Neighborhoods and Protest Message*

In beaconless protocols, the locations of the neighbors are not known in advance, but they are revealed one by one when they reply to the forwarder's request. From a graph theoretic point of view, the candidate nodes are inserted into the set of neighbors, and the insertion order is given by the delay function. This determines the resulting neighborhood, because after one node responds, others may be suppressed and remain

quiet. In order to formalize this mechanism, we introduce the definition of an *ordered neighborhood.*

Let G denote a graph and $\Gamma(u)$ the set of neighbors of a node u in G. For a node u, we define a total order π_u so that $\pi_u(v)$ is the rank of $v \in \Gamma(u)$.

Definition 4: A node $v \in \Gamma(u)$ is hidden, if it is suppressed by a non-hidden node w with smaller rank, i.e. $\exists w \in \Gamma(u)$ with $\pi_u(w) < \pi_u(v)$ and $w \in N(u,v)$.

Definition 5: The π-ordered neighborhood $\Gamma_\pi(u)$ contains all nodes v for which there is no non-hidden node $w \in N(u,v)$.

An ordered neighborhood can be constructed by inserting nodes one by one, if they fulfill the proximity condition (e.g. empty Gabriel circle). In contrast to the original proximity graph, this condition is only checked for the nodes which have been already added to the neighborhood. Note that in contrast to ordered θ-graphs [3], π defines a local order for each node. In BFP a distance-based delay function is used (equation 1) which defines the insertion order and determines the neighborhood. The result of Phase 1 of the BFP algorithm is a distance-ordered neighborhood, which contains at least the edges of the desired subgraph.

Theorem 1: In a proximity graph, the ordered neighborhood of a node v is a superset of the original neighborhood, i.e. $\Gamma_\pi(v) \supseteq \Gamma(v)$.

Proof: Let u be a neighbor of v, i.e. $u \in \Gamma(v)$. Then, the proximity region *N(v, u)* is empty and remains empty, regardless of the rank of u. Thus, $u \in \Gamma_\pi(v)$. ■

When constructing the ordered neighborhood, we can be sure, that the nodes of the desired subgraph are included, but there may be violating edges depending on the insertion order. Therefore, Phase 2 of the BFP algorithm is required, where the hidden nodes send protest messages to indicate edges violating the proximity condition.

4.4 *Distance-ordered neighborhoods*

The worst case number of violating edges depends on the order (i.e. the delay function) and also on the chosen subgraph construction. In case of the Gabriel graph, this number is unbounded, whereas in case of the RNG it is constant.

Theorem 2: A distance-ordered Gabriel neighborhood contains an unbounded number of violating edges.

Proof: The construction in Fig. 4.2 shows that a node can have $\Theta(n)$ neighbors in its distance-ordered Gabriel neighborhood while it has only one valid Gabriel neighbor. Nodes $w_1,...,w_5$ are placed around v with increasing distance and partially overlapping Gabriel circles as shown in the figure. In the Gabriel neighborhood w_1 inhibits an edge (v,w_2), w_2 inhibits an edge (v,w_3) etc., so that v has only one valid edge. In the distance-ordered neighborhood w_1 is inserted first and w_2 is hidden, because node w_1 is in its Gabriel circle. Node w_3 becomes a neighbor, because w_2 is hidden and not part of the neighbor set. Every second node in the chain will become a neighbor of v, i.e. $\Gamma_\pi(v)$ has a size of $\lceil (n-1)/2 \rceil$.

■

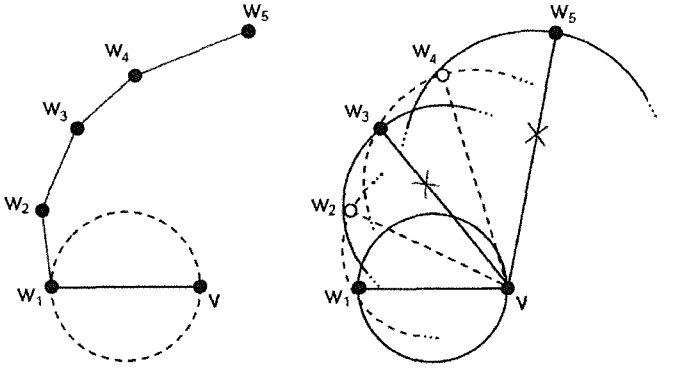


Fig. 4.2. Gabriel graph (left) and distance-ordered neighborhood (right) with hidden nodes (white) and violating edges.

Corollary 1: The beaconless Gabriel graph construction with a distance-based delay function requires an unbounded number of protests in the worst case.

The crucial property to bound the number of protests is that a circular sector has to be part of the proximity region.

Theorem 3: A distance-ordered neighborhood has at most $\lfloor 4\pi/\theta \rfloor - 1$ violating edges, if the proximity region contains a circular sector of angle θ.

Proof: Let $\lhd_\theta(u,v)$ be a sector of the circle $C(u,|uv|)$ with angle θ and $\overline{uv}$ as bisecting line (see Fig. 4.3), and assume that it is contained in the proximity region. A node w is only included in the neighbor set of u, if $\angle vuw > \theta/2$, because of the following reason: if $|uw| < |uv|$, w must be outside $\lhd_\theta(u,v)$. Otherwise, v must be outside $\lhd_\theta(u,w)$. Therefore, we can insert valid neighbors in $\Gamma_\pi(u)$ only at an angular distance of more than $\theta/2$ to an existing neighbor. Then the maximum node degree of v is $\lfloor 4\pi/\theta \rfloor$. This is the limit for the number of violating edges and this limit can be reached in the worst case: the example in the figure shows that for a pair of nodes with overlapping proximity regions there always can be a hidden node x, with higher rank than v and $v \in N(u,x)$ and $x \in N(u,w)$, that renders *(u,w)* a violating edge. ■

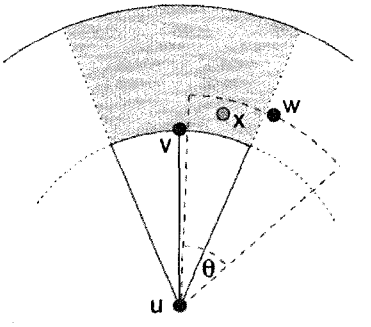


Fig. 4.3. A proximity region containing a sector bounds the number of violating edges.

This theorem shows that we can limit the number of violating edges by choosing an appropriate proximity region. The relative neighborhood graph fulfills this criterion.

Theorem 4: A distance-ordered relative neighborhood contains at most 4 violating edges.

Proof: The RNG lune contains a circular sector of $\theta < 120°$. From this fact and Theorem 3 follows the result. ■

Corollary 2: The beaconless RNG construction with a distance-based delay function requires a constant number of protests in the worst case.

However, the proximity region of the RNG is quite large, such that more edges are forbidden than in the Gabriel graph. The RNG has (length/power) stretch factor $\Theta(n)$, the Gabriel Graph only $\Theta(\sqrt{n})$ (both are not hop-spanners) [2].

4.5 Relevance of Protest Messages

We have seen that in the presence of hidden nodes edges can be created that violate the proximity condition. Therefore it is necessary to allow hidden nodes to protest against the selection of a neighbor. One might ask if there is any delay function or any practical subgraph construction that favors only the valid neighbors. Unfortunately this is not the case.

Theorem 5: No undirected, planar and connected proximity graph can be constructed without protests.

Proof: Consider the scenario in Fig. 4.4 as a counterexample. Node w is located in the suppression region of v, v is suppressed by u, but w is not suppressed by u. When considering the suppression region for arbitrary proximity graphs (that are undirected, planar and connected), the region is at least the suppression region of the Gabriel graph and at most the suppression region of the RNG. This follows from Lemmas 1 and 2. Therefore, region A is part of the suppression region of v and region B is not a suppression region of u for all considered proximity graphs. Now we build the ordered-neighborhood of x for all permutations of u, v, and w.

Insertion order π; (·) denotes hidden node	Neighborhood $\Gamma_\pi(x)$	Immediate protest	Protest of hidden nodes
u (v) w	*{u, w}*		*v*
u w (v)	*{u, w}*		*v*
v u (w)	*{u, v}*	*u*	
v (w) u	*{u, v}*	*u*	
w u (v)	*{u, w}*		*v*
w v u	*{ }*	*v, u*	

We can see from the table, that regardless of the insertion order there is always a protest, either because the inserted node immediately knows that it violates the proximity graph condition, or because of a hidden node that protests later. ■

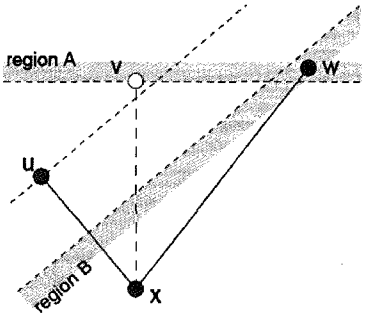


Fig. 4.4. Hidden node scenario for Theorem 5.

Chapter 5 Circlunar Neighborhood Graph

For the beaconless subgraph construction we want to preserve as much edges as possible, bound the number of protests and obtain a planar graph. The planarity can be achieved by including the Gabriel circle in the proximity region. Protests can be bounded by including a circular sector. The larger the angle of the sector, the smaller the maximum node degree, but this also cancels more edges. Therefore, we propose the Circlunar Neighborhood Graph (CNG) as an alternative to Gabriel graph and RNG. It is a planar graph with constant degree; its proximity region is only a small enhancement of the Gabriel circle and the proximity condition can be tested with 1-hop knowledge and simple arithmetics. Notice that in this thesis we only consider the planar graphs which can be constructed in a localized manner, i.e. those which only require 1-hop knowledge.

Definition 6: The circlunar neighborhood $N_{CNG}(u,v)$ of two points u and v is given by the intersection of four disks of radius $|uv|$ centered at the corners of a square of which (u,v) is the diagonal (see Fig. 5.1).

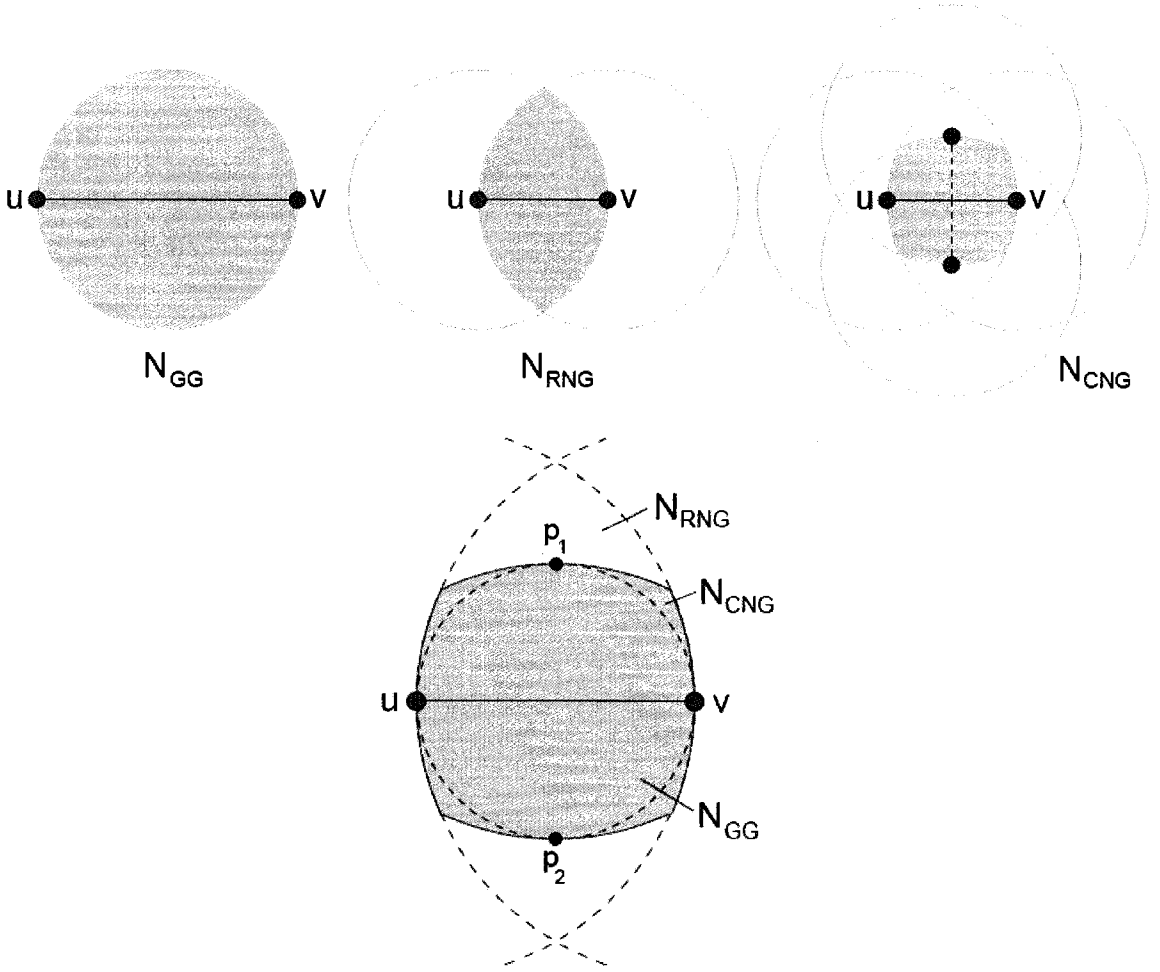


Fig. 5.1. The circlunar neighborhood with RNG lune and Gabriel circle.

The circlunar neighborhood graph contains an edge *(u,v)* if and only if $N_{CNG}(u,v)$ is empty:

Definition 7: The circlunar neighborhood graph of a node set V contains an edge *(u,v)* iff $|uv| \le \max\{|uw|,|vw|,|p_1w|,|p_2w|\}$, $\forall w \in V, w \neq u,v$.

5.1 Properties of the Circlunar Neighborhood Graph

The CNG has a strong relation to Gabriel graph and RNG and inherits planarity and connectivity.

Theorem 6: The circlunar neighborhood graph of a node set V is planar and connected, if the unit disk graph of V is connected.

Proof: Follows from the shape of the proximity region and Lemmas 1 and 2. ■

The CNG also inherits a disadvantage from the RNG, namely the unbounded spanning ratio of $\Theta(n)$ (the maximum ratio of shortest path in CNG over shortest path in the original graph). One can construct the same lower bound example ("RNG tower" [2]) for the CNG. In other words, when using the CNG planarization, the maximum detour is unbounded in the worst-case. Apart from these worst-case considerations, we performed simulations on 200 random unit disk graphs with 100 nodes for network densities (average number of neighbors) between 4 and 12. Measurements of the spanning ratio show that the CNG is closer to the Gabriel graph than to the RNG: The hop spanning ratio of the CNG is only 5%-7% larger than in the Gabriel graph, while the RNG's spanning ratio is 36%-61% larger. Assuming the uniform distribution or the graph nodes in a square, the CNG has an expected node degree of 3.6 and is thus sparser then the Gabriel graph and denser than the RNG. Table 5.1 summarizes these results (see [9]).

Graph	Expected degree [9]	Maximum degree	Spanning ratio [2]
RNG	2.558	5	$\Theta(n)$
CNG	3.598	14	$\Theta(n)$
GG	4.000	$n-1$	$\Theta(\sqrt{n})$

Table 5.1: Properties of RNG, CNG and GG

Following the considerations in [9], we can derive the expected degree of the CNG from the ratio of the circle $C(u,|uv|)$ and the area A of the proximity region $N_{CNG}(u,v)$. The area of the circlunar neighborhood is $A \approx 0.873r^2$. This gives an expected degree of $C(u,|uv|)/A \approx 3.598$.

5.2 *Beaconless construction*

The CNG enables a beaconless planar subgraph construction with a constant number of protests as the following theorem shows.

Corollary 3: A distance-ordered neighborhood in the CNG has at most 13 violating edges.

Proof: This follows from Theorem 3. One can show that the circlunar neighborhood contains a circular sector of $\approx 48.6°$. Plugging this into Theorem 3 gives the result. ■

5.3 *Face Routing on the Circlunar Neighborhood Graph*

The circlunar neighborhood graph has the structural graph properties that are necessary to guarantee recovery. The following graph property holds for the Gabriel graph (Lemma 1 in [12]) and can be shown analogously for the CNG.

Lemma 3: For any edge (u,v) crossing the *s-t* line connecting source s and destination t in the circlunar neighborhood graph, at least one of the end points u or v is closer to the target than s.

Proof: As the circlunar neighborhood contains the Gabriel circle, the Gabriel circle over (u,v) contains neither s nor t. It follows that $\angle usv$ and $\angle utv$ are less than $\pi/2$. Since the sum of the angles of the quadrangle $usvt$ is 2π, at least one of the angles $\angle sut$ or$\angle svt$ is greater than $\pi/2$. This implies that at least one of the nodes u or v is closer to t than s. ■

For guaranteed delivery, face routing on the planar subgraph has to provide progress towards the destination. This is shown by the following theorem (see Corollary 2 in [12]).

Theorem 8: Let s and t be nodes in a circlunar neighborhood graph. When starting at s, face routing will always find a node v that satisfies $|vt| < |st|$.

Proof: The CNG is planar and from Lemma 5 in [12] follows that face routing will always find an edge intersecting the *s-t* line. With Lemma 3 we can conclude, that one of the edge's end points satisfies $|vt| < |st|$. ■

Chapter 6 Angular Relaying

6.1 Angular Relaying using a Sweep Line

Angular relaying is a beaconless face routing strategy, which can be used as a method for recovery from local minima. While BFP works independent of the routing protocol, angular relaying needs the information of the previous hop and the recovery direction (right-hand or left-hand). It utilizes an angle-based delay function to determine a candidate for the next hop which is used in combination with the select-and protest method for avoiding crossing edges. Here, we use the Gabriel graph condition as planarization criterion.

By using an angle-based delay function the first neighbor in counter-clockwise order is selected. Other approaches, such as NB-FACE, the clockwise relaying approach in an earlier version of BLR [15], or the Bypass method of PSGR are also based on an angle-based function, but they either cannot guarantee delivery or the complete neighborhood is involved in the message exchange. A simple angle-based delay function has the following form:

$$t(\theta)=\frac{\theta}{360^\circ}t_{\max} \tag{2}$$

The angle θ can be considered in clockwise or counterclockwise order, depending on the traversal direction (left-hand or right-hand). Selecting a candidate by this function is not sufficient to guarantee delivery, because it is not necessarily a neighbor of the forwarder in the Gabriel subgraph. Therefore, we use protest messages to prevent crossing links. This is similar to the protest phase used in the BFP algorithm. The angular relaying algorithm consists of the following two phases.

Selection phase After receiving a packet from the previous hop u, the forwarder v sends an RTS (including previous hop u and its own position) and sets its timer to $t_{\max}$. Every candidate w sets its timer $t(\theta)$ using the angular distance $\theta = \angle uvw$ to the previous hop (see equation 2). Candidates answer with CTS in counter-clockwise order

according to the delay function. We allow candidates to respond, if they have the previous hop in the Gabriel circle (i.e. nodes in region B in Fig. 6.1). These nodes answer with an "invalid CTS", because they violate the Gabriel graph condition, but other nodes should be aware of their existence. Otherwise they would be hidden and would need a chance to protest later. After the first candidate w answers with valid CTS, the forwarder immediately sends a SELECT message announcing that w is the first selected node. All candidates with pending CTS answers cancel their timers.

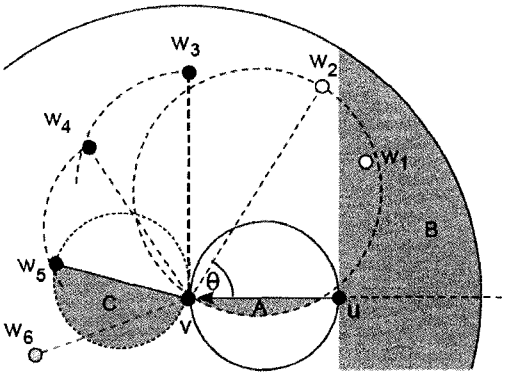


Fig. 6.1. Angular Relaying: w_1 and w_2 are invalid, w_3 is selected, w_4 and w_5 protest; w_5 is the next hop.

Protest phase After the selection of the first candidate, the protest phase begins. The forwarder v starts its protest timer that covers only the time when protests can occur, which is

$$t_{pr} = t\left(\frac{\pi}{2}\right) = \frac{1}{4}t_{\max} \tag{3}$$

for the Gabriel graph. Now, no further CTS answers are allowed. Instead, each candidate x sets a new timer

$$t(\theta'), \text{ where } \theta' = \angle uvx - \angle uvw \tag{4}$$

according to equation 2, which determines the order of protests. First, only nodes in $N_{GG}(v,w)$ are allowed to protest. If a node x protests then it automatically becomes the next hop. After that, only nodes in $N_{GG}(v,x)$ are allowed to protest. Finally, if the forwarder's timer expires (i.e. there are no more protests), the data packet is sent to the currently selected (first valid or last protesting) candidate.

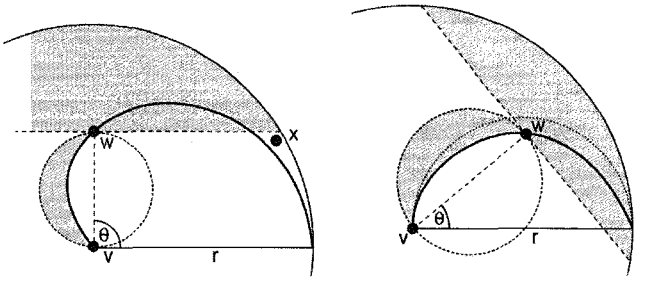
Angular Relaying using a simple angle-based delay function (equation 2) is similar to NB-FACE. In NB-FACE the forwarder waits for a time span τ after the first candidate responded, in order to leave room for protests (*Nack*). After that, it sends a message (*Fin*) to stop the contention period and select the final candidate. If τ is a constant angle, then the case of cascading protests is not covered; otherwise, if τ spans the whole rotation, then all neighbors respond, even if they are not protesting, and the advantage over the Request-Response approach vanishes. Also the details about how nodes are treated that have the previous hop in their Gabriel circle (region B in Fig. 6.1) are left open.

6.2 Angular Relaying using a Sweep Curve

The contention process using the angular delay function can be regarded as a rotating sweep line, i.e. a ray from v through u (Fig. 6.1) that rotates in counter-clockwise order until it hits the first node w. This node will be the next hop, if it is a valid neighbor so far and there are no protests afterwards. Protests are issued by nodes that lie in the Gabriel circle over (v,w) and beyond the sweep line (region C in Fig. 6.1). Therefore, it makes no sense to use CNG or RNG with Angular Relaying, because the area of possible protesting nodes would be even larger. We also observe that the protest area grows with the distance of a candidate to the forwarder.

This leads to the question whether another shape of the sweep line could be applied such that closer nodes may respond earlier and the area of possible protesting nodes is reduced. To ensure that the most suitable nodes respond first, the sweep curve must have the following property:

Sweep curve property: For a node w on the sweep curve the following must hold: if there is another node x ahead of the sweep curve in counter-clockwise order, then either $\angle uvw < \angle uvx$ or x is not a Gabriel neighbor of v, where u is the previous packet holder.



(a) (b)

Fig. 6.2. Angular relaying with different sweep curves.
(a) When using an arbitrary spiral, there can be a valid Gabriel neighbor x beyond the sweep curve with smaller angle θ and larger delay.
(b) The optimum Archimedean spiral that fulfills the sweep curve property.

In other words, if w responds first, then there are no other Gabriel neighbors with smaller angular distance θ to the previous hop u, which could respond later and contradict w being the first neighbor in counterclockwise order (see Fig. 6.2). In order to determine a valid sweep curve (for the Gabriel graph construction), we consider the suppression region for a node w and calculate the positions for which all nodes with smaller θ are suppressed (see Fig. 6.3). Fulfilling the sweep curve property requires that $x + z \leq r$. For the height of a rectangular triangle holds $y^2 = xz$ and it follows that

$$x + \frac{y^2}{x} \leq r,$$

i.e. all nodes on the sweep curve should lie between the straight line and the semi-circle in Fig. 6.3.

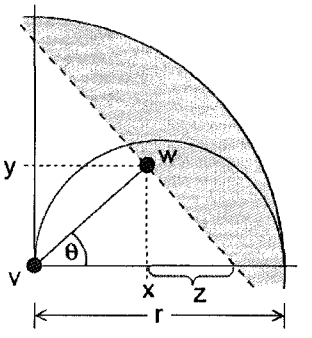


Fig. 6.3. Suppression region and sweep curve property

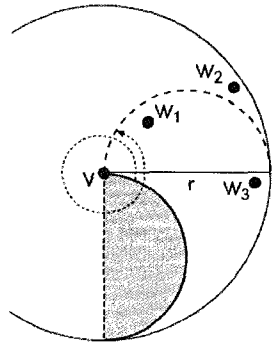


Fig. 6.4. The general sweep curve needs a $5\pi/2$ turn to cover all nodes

6.3 *Correctness of the Angular Relaying Algorithm*

Theorem 9: The angular relaying algorithm selects the first edge of the Gabriel subgraph in counter-clockwise order.

Proof: Let $N^{+}_{GG}(v,w)$ be the left part of the Gabriel circle of (v,w), which is ahead of the sweep line/curve (region C in Fig. 6.1). Analogously, let $N^{-}_{GG}(v,w)$ be the remaining part of the Gabriel circle. For the first selected candidate w the $N^{-}_{GG}(v,w)$ is empty due to the following reasons. First, all nodes are allowed to respond, including the invalid ones. This ensures that there are no hidden nodes invalidating w. Thus, w has the smallest angle $\angle uvw$ among the Gabriel neighbors (otherwise another valid neighbor would have responded before). There is only one region that we did not consider yet, namely the part of $N^{-}_{GG}(v,w)$ beyond uv (region A in Fig. 6.1). But this region is empty, because it is always covered by $N_{GG}(u,v)$; otherwise, (u,v) would not be an edge of the Gabriel graph. For similar reasons, $N^{-}_{GG}(v,w)$ is empty for nodes that protest in the second phase. Protesting nodes are automatically selected as tentative next hop. Thus for the currently selected node w holds the *invariant* of the algorithm: If $N^{+}_{GG}(v,w)$ is empty, then w is the Gabriel neighbor of v with the smallest angle $\angle uvw$. This follows from the sweep curve property (no node with smaller angle responds later) and the considerations above. The algorithm terminates if the forwarder's timer expires. Then $N^{+}_{GG}(v,w)$ is empty because there are no further protests. If a part of the Gabriel circle intersects with the radial line

from v through u, then this part lies within the Gabriel circle over (u,v) and therefore, this region is also empty. ■

6.4 *Sweep Curve Functions*

In general, a sweep curve function f, which describes a point on the sweep curve by angle θ and distance $f(\theta)$, has to be monotonic and fulfill the following conditions:

(1) $f(0)=1$;

(2) $f(\pi/2)=1$;

(3) $0 \le f(\theta) \le \cos(\theta)$ for $\theta \in [0,\pi/2]$.

The delay function is derived from the inverse of f and has the following general form:

$$t(d,\theta)=\frac{\theta - f^{-1}(d/r)+\pi/2}{5\pi/2} t_{\max} \tag{5}$$

In order to calculate the expected protest area, we consider a fixed angle θ and calculate the area $A_P(\theta)$ enclosed by the sweep curve and the Gabriel semi-circle (shaded lune in Fig. 6.2b).

$$A_P(\theta)=\frac{1}{2}\pi\left(\frac{f(\theta)}{2}\right)^2 - \int_{\theta}^{\frac{\pi}{2}}\int_{0}^{f(\theta)} r\,dr\,d\varphi = \frac{\pi}{8}f(\theta)^2 - \frac{1}{2}\int_{\theta}^{\frac{\pi}{2}} f(\varphi)^2\,d\varphi \tag{6}$$

The probability of a node in distance r is $2\pi r/\pi = 2r$ in the unit circle. Thus, the expected protest area is given by

$$E[A_P]=\int_0^1 A_P\left(f^{-1}(r)\right) 2r\,dr \tag{7}$$

A logarithmic spiral is not a valid sweep curve, because it violates the sweep curve property (see Fig. 6.2a). Valid sweep curves are the semi-circle ($f(\theta)=\cos(\theta)$) or some Archimedean spirals. An Archimedean spiral has the general form $a+b\theta^c$. We use the form $f(\theta)=1-(\frac{2}{\pi}\theta)^c$, which fulfills conditions 1 and 2. The exponent c determines the shape of the curve. From

$$1-(\tfrac{2}{\pi}\theta)^c \le \cos(\theta) \text{ follows that } c \le \frac{\log(1-\cos(t))}{\log(\frac{2}{\pi}\theta)} \tag{8}$$

i.e. $c \leq 1.56$ for $\theta \in [0, \pi/2]$. Using equation 7, the expected protest area is minimized, if $c \approx 1.259$ (numeric evaluation).

With this function, we can reduce the expected protest area by more than a factor of 2 compared to the sweep line. Table 6.1 summarizes the results for different functions.

Sweep curve function	$f(\theta)$	$f^{-1}(r)$	Expected protest area
sweep line	—	$\pi/2$	0.1963
semi-circle	$\cos(\theta)$	$\arccos(\theta)$	0.0982
Archimedean spiral	$1-(\frac{2}{\pi}\theta)^2$	$\frac{\pi}{2}(1-d)^{1/c}$	0.0897
logarithmic spiral*	$\exp(-\frac{2}{\pi}\theta)$	$\frac{\pi}{2}\ln(d)$	(0.0531)
*) not a valid sweep curve			

Table 6.1. Sweep curve variants

6.5 *Angular Relaying: Algorithm Pseudo Code*

Suppose *G=(V,E)* is the unit disk graph, *C(u)* is the list of all invalid CTSs received by node *u* and *P(u)* is the list of Protests received by node *u*. The task of face routing with Angular Relaying is to deliver the packet from the node where greedy routing failed to either the recovery or destination node.

Algorithm 4. Face routing with Angular Relaying, forwarder node

```
1:  Let u be the current forwarder or the packet
2:  Let m be the local minimum distance to destination
3:  repeat
4:      u sends RTS (including previous hop and its own position)
5:      u sets timer T1 to tmax and clears C(u)
6:      for each CTS received by u from node v
7:          if CTS is "valid" *
8:              u sends SELECT message including v's position and its own position
9:              u creates empty list P(u) and adds node v to P(u)
10:             u stops timer T1 and breaks for loop
11:         else
12:             u adds v to C(u)
13:         end if
14:     end for
15:     after expiry of timer T1 u sets timer T2 to tpr according to equation 3
16:     for each Protest from node x received by u
17:         u adds x to P(u)
18:     end for
19:     when T2 expires u sends the packet to the last node in P(u), i.e. u=P(u)last,
20: until packet delivered to destination or u is closer to destination than m
```

*) CTS from node v is considered valid is if $v \notin N_{GG}(u,x)$ for any $x \in C(u)$, where $N_{GG}(u,x)$ is a proximity region of Gabriel graph over (u,v).

Now suppose $N(u)$ is the neighborhood of node u and $P(v)$ is the list of protests overheard by node v.

Algorithm 5. Face routing with Angular Relaying, neighbor node

```
1:  for each node v in N(u)
2:      v sets timer T1 to t(θ) according to equation 2 (or equation 5 for sweep curve)
3:      (let s be the next hop candidate SELECTed by the forwarder u)
4:      after expiry of timer T1 and if s is not yet available v sends CTS to u
5:      v sets timer T2 to t(θ') according to equation 4
6:      for each Protest from node x received by v
7:          v adds x to P(v)
8:      end for
9:      after expiry of timer T2
10:     if P(v) is empty then
11:         v sends Protest to u with its own position
12:     else
13:         v sends Protest to u only if v ∈ N_GG(u, P(u)_last)
14:     end if
15: end for
```

6.6 *Angular Relaying With Sweep Curve: Example*

Consider the following example.

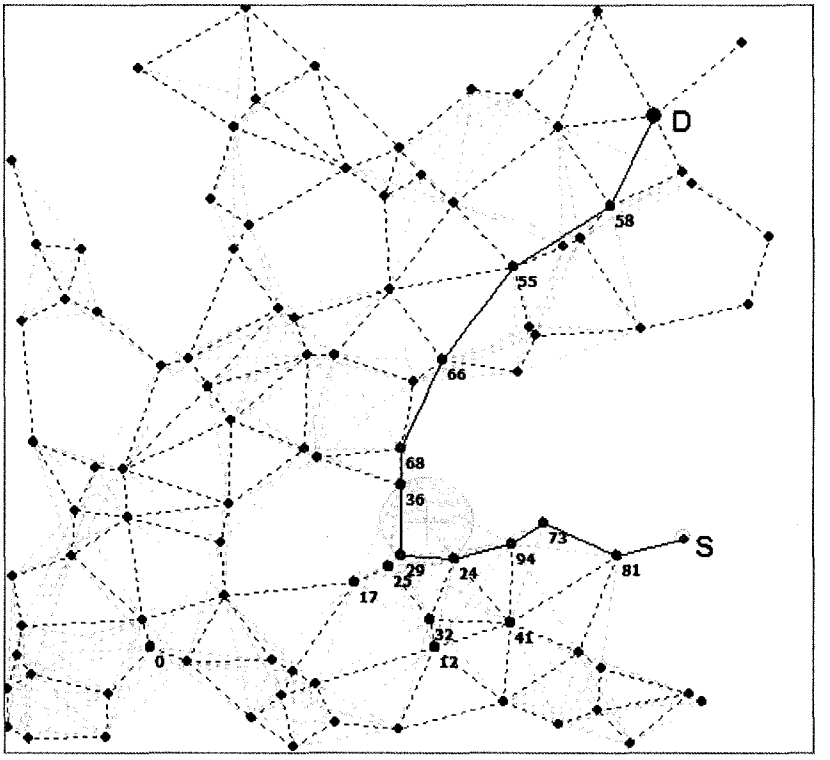


Fig. 6.5. Example or Angular relaying with sweep curve

The task is to deliver the packet from node S to destination D using Greedy-Face-Greedy routing. In the recovery mode the Angular Relaying algorithm with sweep curve $f(\theta) = \cos(\theta)$ (semi-circle) should be used.

The path produced by the algorithm is S-81-73-94-24-29-36-68-66-55-58-D as shown in the figure above. The greedy routing fails right away in node S and face routing is initiated. There are two recovery nodes along the way, 73 (recovery complete, but there is no greedy choice available) and 68 (recovery complete).

Let's consider what happens in node 24 in greater detail. Node 24 receives the packet from node 94. It then starts the selection phase of Angular Relaying by broadcasting the RTS message. Each neighbor of node 24 starts the CTS timer T_1 according to equation 5, where $f^{-1}(d) = \arccos(d)$. Here is the list of neighbors sorted by timeout:

Node Id	73	36	68	29	25	17	32	12	41	94
Angle	8.5	109.6	100.0	162.7	174.2	180.9	235.4	244.1	294.2	360.0
Timeout	0.12	0.34	0.39	0.41	0.45	0.51	0.59	0.64	0.74	0.86

Neighbour 73 replies with CTS first. This CTS is "invalid" because of node 94 and is discarded. Neighbour 36 sends the next CTS, and it is valid. Forwarder 24 ends the phase 1 by broadcasting a SELECT message with position of node 36.

Now the protest phase begins. Forwarder 24 sets its second timer T_2 to t_{max}. All remaining neighbours cancel their CTSs and start the protest timer T_2 according to equations 4 and 5. The sorting of neighbours by timeout remains the same as above. Node 68 is the first one to protest, but it lies outside the allowed protest area; thus it remains silent. Node 29, however, lies inside the protest area; it sends protest with its own position to forwarder 24. Node 24 eliminates node 36 and records node 29 as the new possible candidate for packet transfer. The rest of the nodes lie outside the protest area and remain silent. As the timer expires and since there are no more protests, the packet is sent to node 29.

Chapter 7 Simulations

We performed simulations of BFP and Angular Relaying on 500 random graphs with 100 nodes for network densities (i.e. average number of neighbors) ranging from 4 to 12. Messages are sent from the leftmost to the rightmost node using GFG routing. The greedy part is performed by a beaconless greedy scheme using RTS/CTS, the face routing part is performed by BFP on different subgraphs or by Angular Relaying using sweep line and sweep curve. We use an ideal MAC layer model assuming uniform transmission radii and no collisions. We measure the number of messages used for each route. In order to obtain a fair and consistent measure for different routing paths and subgraphs, the values are normalized, i.e. divided by the length (number of hops) of the shortest path. All the results are shown using the error bars diagrams using the 95% confidence interval.

7.1 Connected Unit Graph Generation

The graphs are generated using Maximum Degree Proximity Algorithm (MAX-DPA) [28]. This section describes this algorithm.

The task is to generate random unit graph with N nodes so that the average density, i.e. number of neighbors per node, is d. Unit graph is defined by $G=(V,E)$, where V is the set of nodes, $E=\{(u,v):|uv|\le r; u,v\in V\}$ is the set of edges, and value r is the transmission radius.

According to MAX-DPA the first node randomly obtains its x coordinate and y coordinate in an interval $[0,a)$. At round k, $2 \le k \le N$, first a random position is generated in the $[0,a)\times[0,a)$ square. To be accepted, the position has to pass the proximity test and the maximum degree test. To pass the proximity test the new position 1) should be closer than r to at least one existing nodes and 2) should be no closer than d_0 to any of the existing nodes, where

$$r = \sqrt{\frac{da^2}{(N-1)\pi}}$$

is an approximate radius which estimates the final transmission radius in the random unit graph, and d_0 is the minimum distance allowed between any two nodes. To pass the maximum degree test the approximate degrees, i.e. currently considered degrees, of node k and all other previously accepted nodes are calculated assuming that node k was placed to this new position. If none of these degrees is greater than or equal to the maximum degree allowed d_{max}, the position is accepted. Otherwise it is rejected and a new position is generated.

After N nodes were generated, all $N(N-1)/2$ potential edges in the network among the N nodes are sorted by their length in the ascending order. The transmission radius r that corresponds to a chosen value of d is equal to the length of $Nd/2$-th edge in the sorted list of edges. Any edge with length less than or equal to chosen r remains in the graph. All other edges are eliminated from the graph. The resulting graph is then checked for connectivity using Dijkstra's shortest path algorithm.

Studies in [28] show that MAX-DPA is approximately 25% faster than the standard connected random unit disk graph generation algorithm for the medium density graphs (d=8) and approximately 4.6 times faster for the small density graphs (d=5).

7.2 Beaconless Forwarder Planarization

We simulated BFP algorithm in 500 random unit disk graphs of various densities. The execution of the algorithm was triggered in every node of each graph. Our simulation show the superior performance in case of the GG and CNG graphs as compared to the RNG. The graphs below summarize the results.

In terms of the number of the average number of CTSs (Fig. 7.1), average number of hidden nodes (Fig. 7.2), number of violating edges (Fig. 7.3) and, as a result, the number of protests (Fig. 7.4) the BFP performance on CNG is closer to the Gabriel graph than the RNG. The CNG offers a significant reduction in the message complexity as compared to GG, while having the density that is close to GG (see Fig. 7.5).

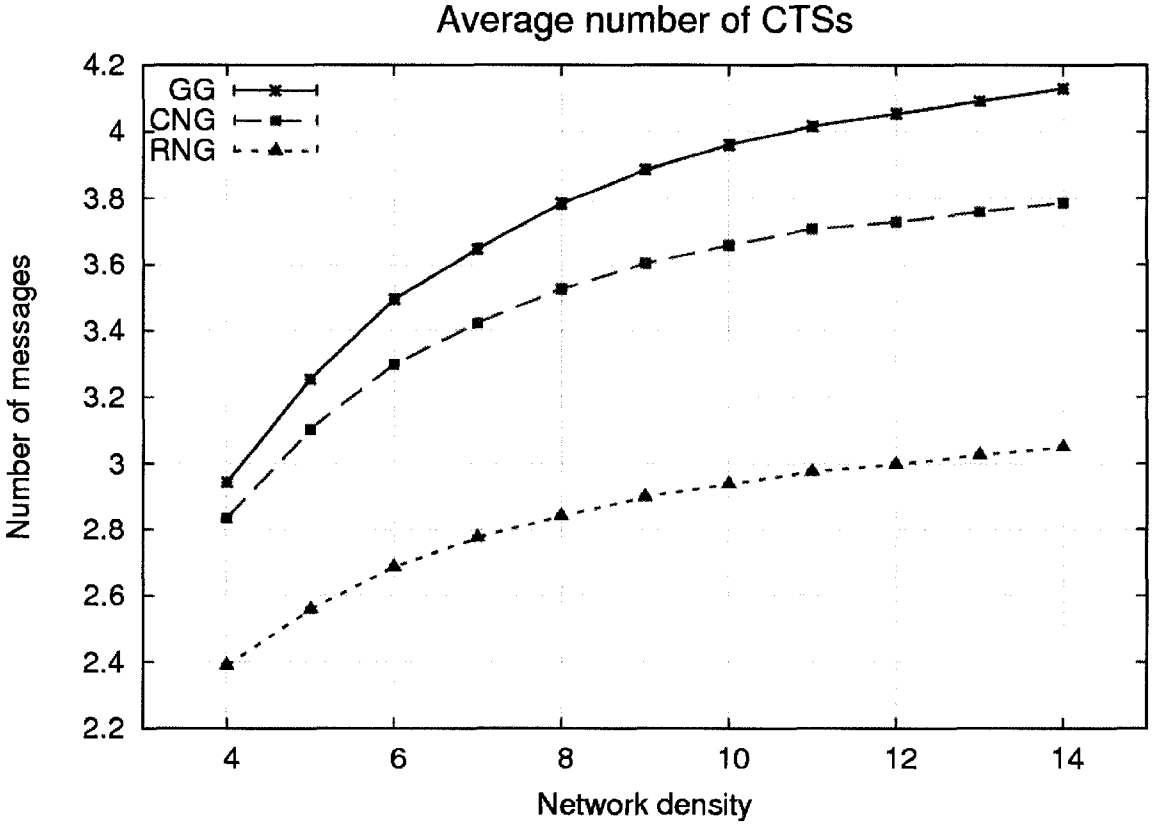


Fig. 7.1. BFP: Average number of CTS messages in BFP

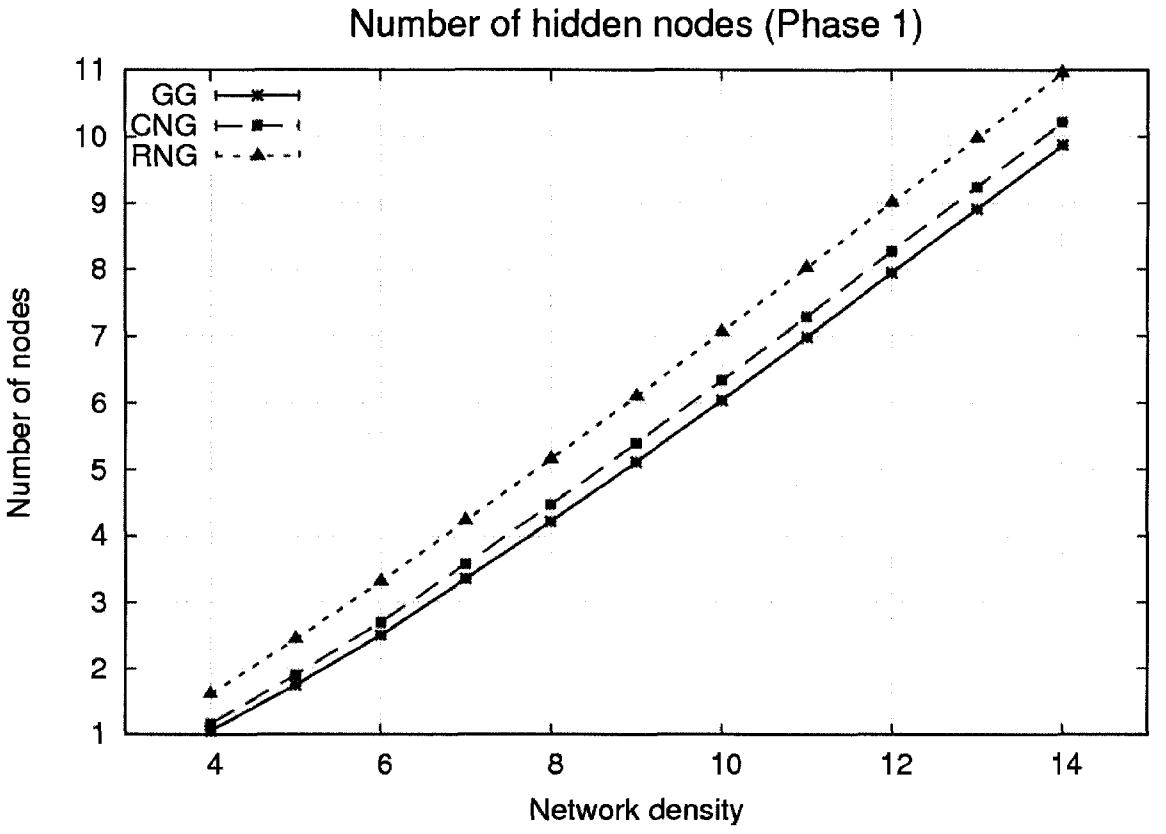


Fig. 7.2. BFP: Average number of hidden nodes in BFP Selection phase

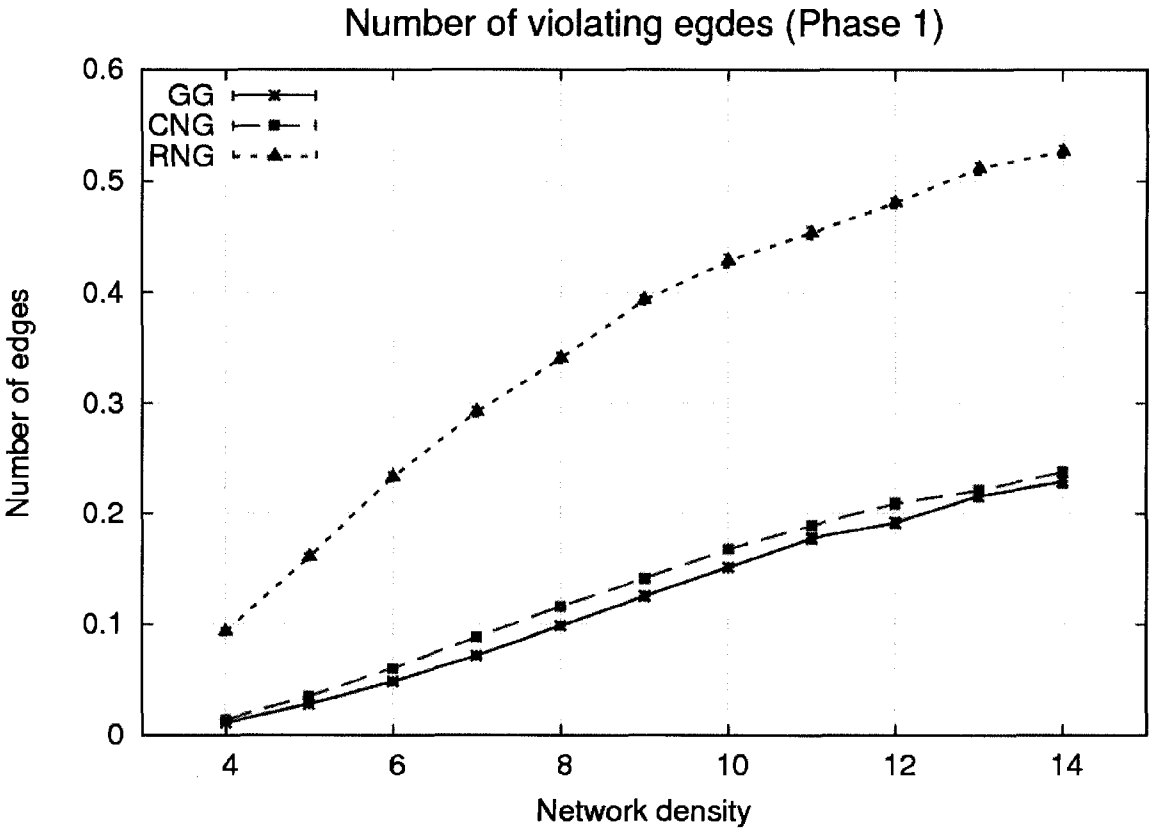


Fig. 7.3. BFP: Average number of violating edges in BFP

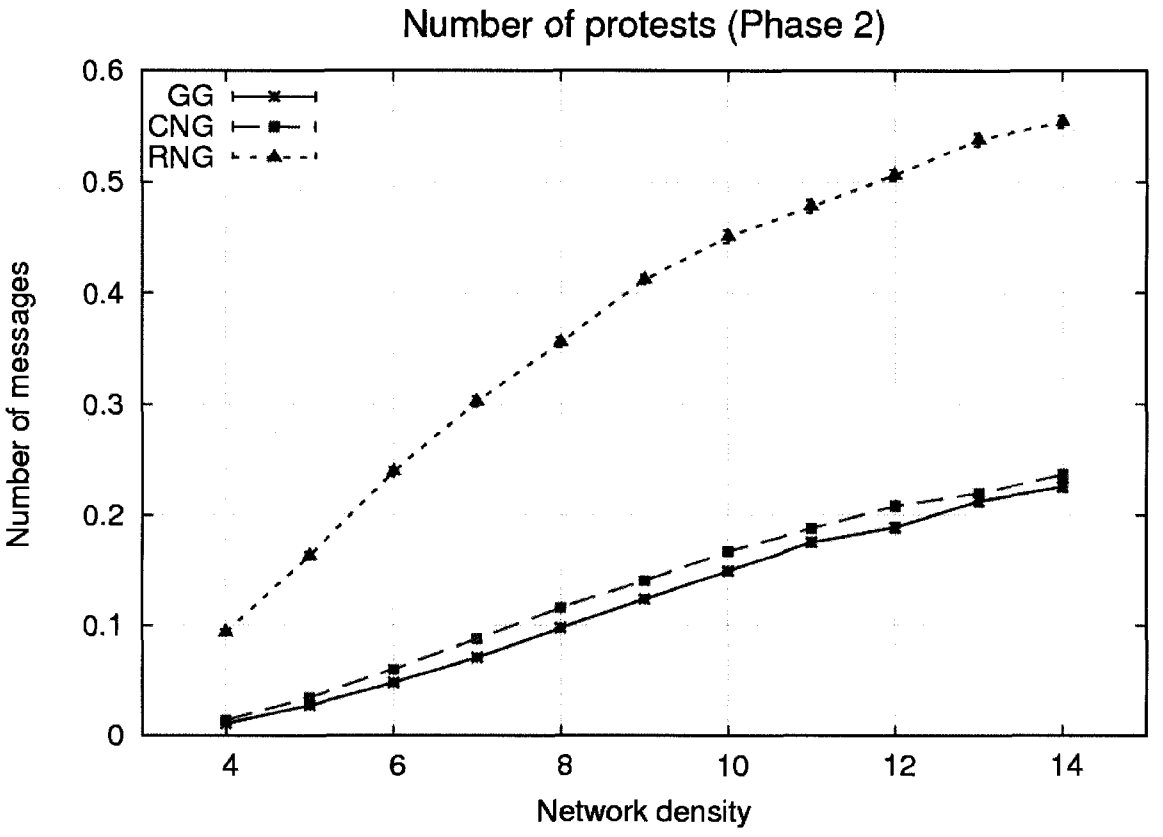


Fig. 7.4. BFP: Average number of Protest messages in BFP

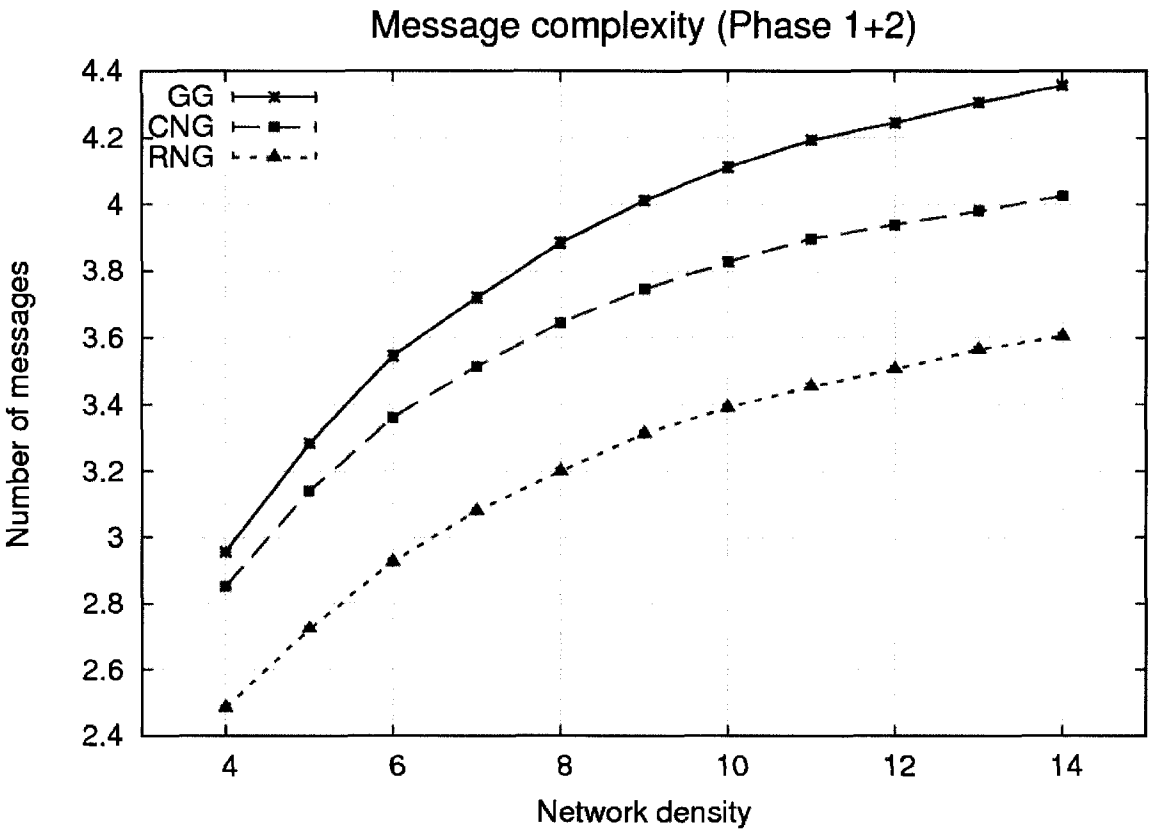


Fig. 7.5. BFP: Average overall message complexity of BFP

7.3 Georouting with Beaconless Forwarder Planarization

The results for the number of protests (Fig. 7.7) and for the overall message complexity (Fig. 7.8) show a gap between Gabriel graph and RNG. The inferior performance of RNG is due to the long detours caused by this planarization method. The CNG reaches the good performance of the Gabriel graph, while guaranteeing a worst-case bound for the number of protests, which is not possible when using the Gabriel graph planarization.

All the results are shown using the error bars diagrams using the 95% confidence interval that summarize simulations on 500 different random UDGs with 100 nodes each for every density value shown.

The results show that the performance of BFP routing on CNG is better than the performance on GG in the number of protests (Fig. 7.7). This causes the better overall message complexity of BFP routing on CNG as compared to GG (see Fig. 7.8). Moreover, our simulations show that the maximum message complexity values for the CNG are very close to the values for GG and are always less than RNG values (Fig. 7.9). It is clear that the BFP routing always produces longer paths for RNG (Fig. 7.11) and initiates the recovery more often on RNG which leads to longer face mode paths (Fig. 7.12). As a result we can state that the CNG graph offers an advantage in message complexity as compared to Gabriel graph, while producing shorter paths than the RNG.

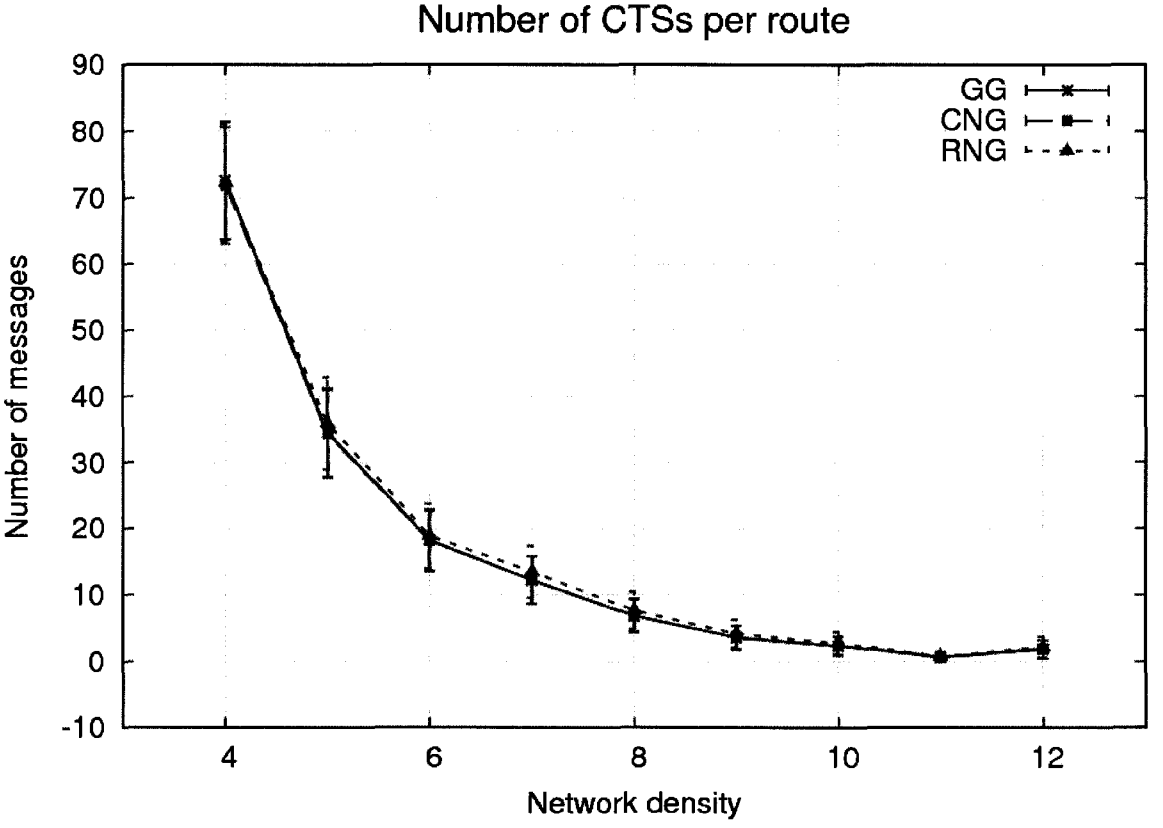


Fig. 7.6. Routing with BFP: Average number of CTS messages per route

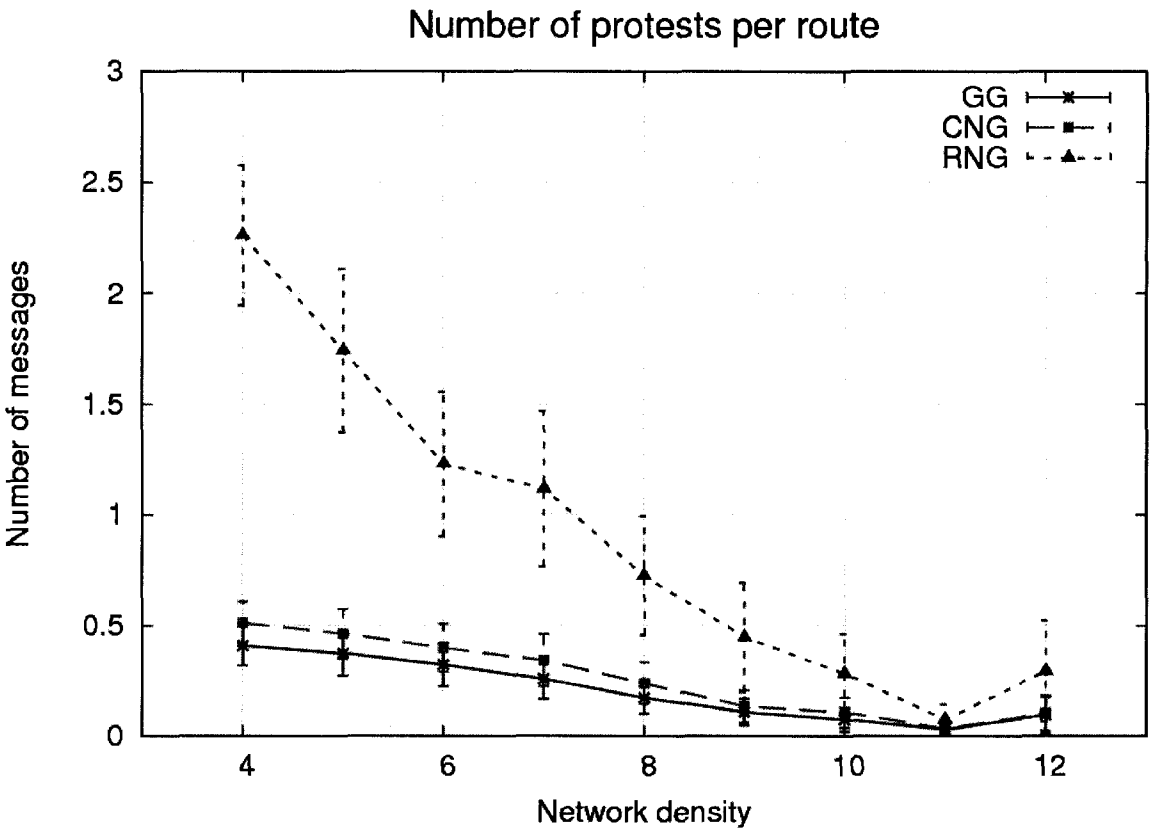


Fig. 7.7. Routing with BFP: Average number of Protest messages per route

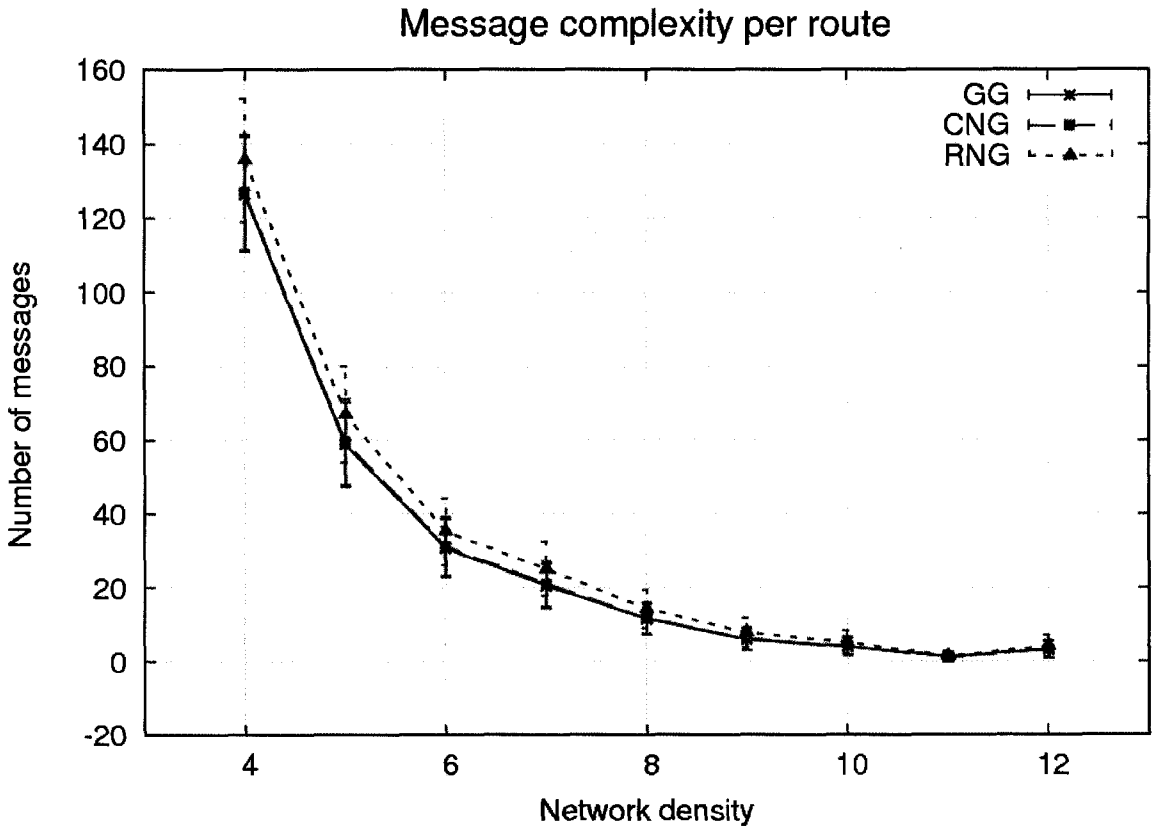


Fig. 7.8. Routing with BFP: Average overall message complexity per route

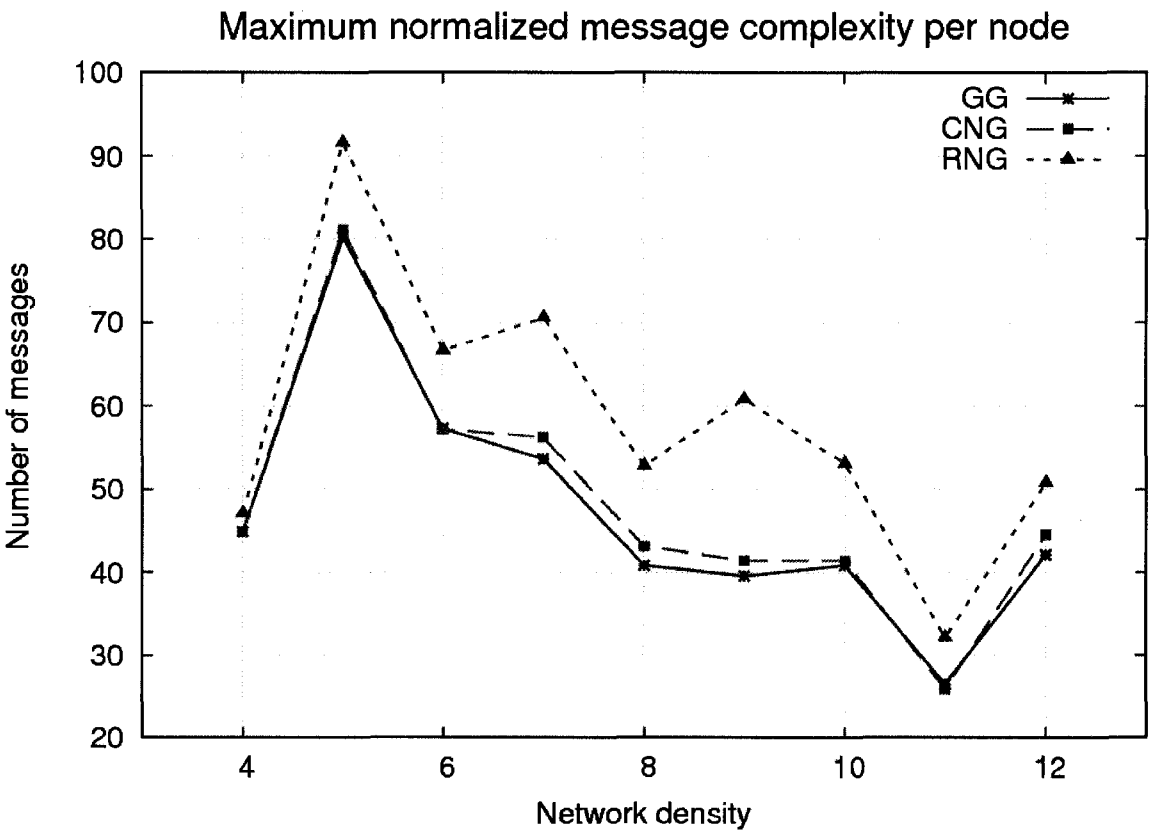


Fig. 7.9. Routing with BFP: Maximum message complexity per node in route

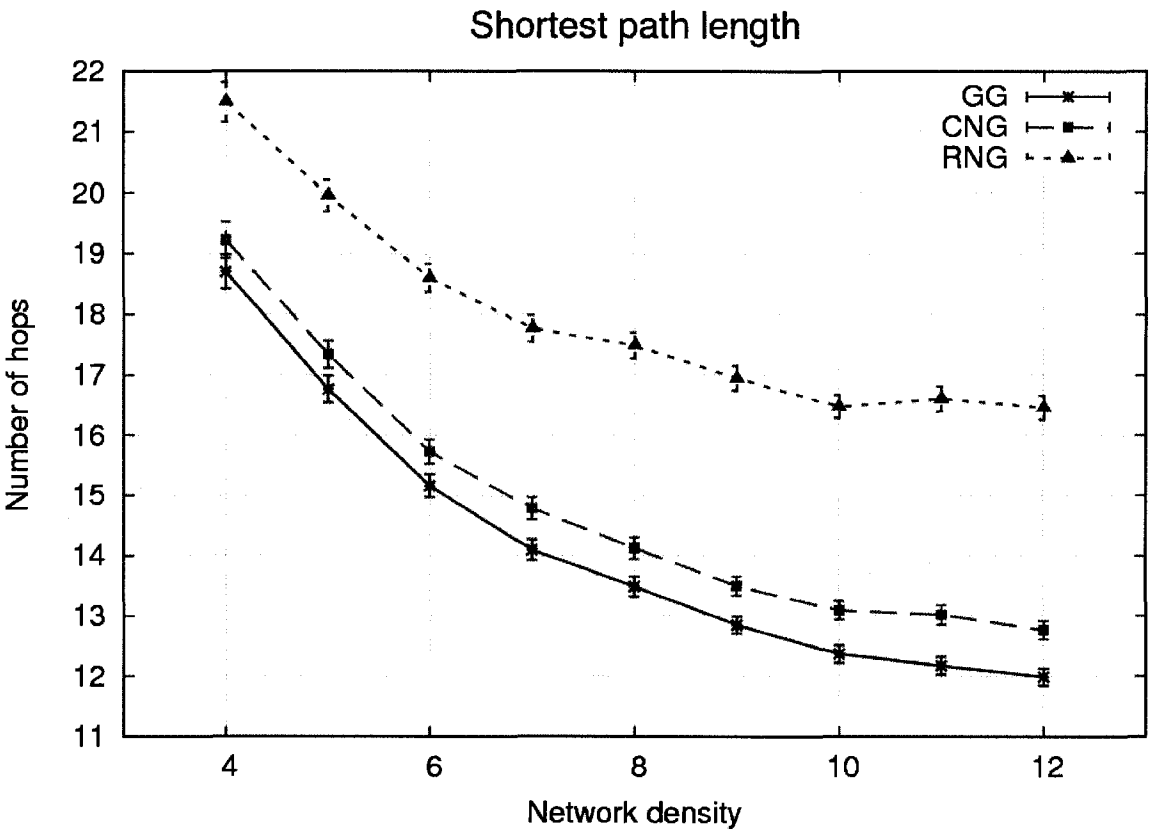


Fig. 7.10. Routing with BFP: Average shortest path length in planar graphs

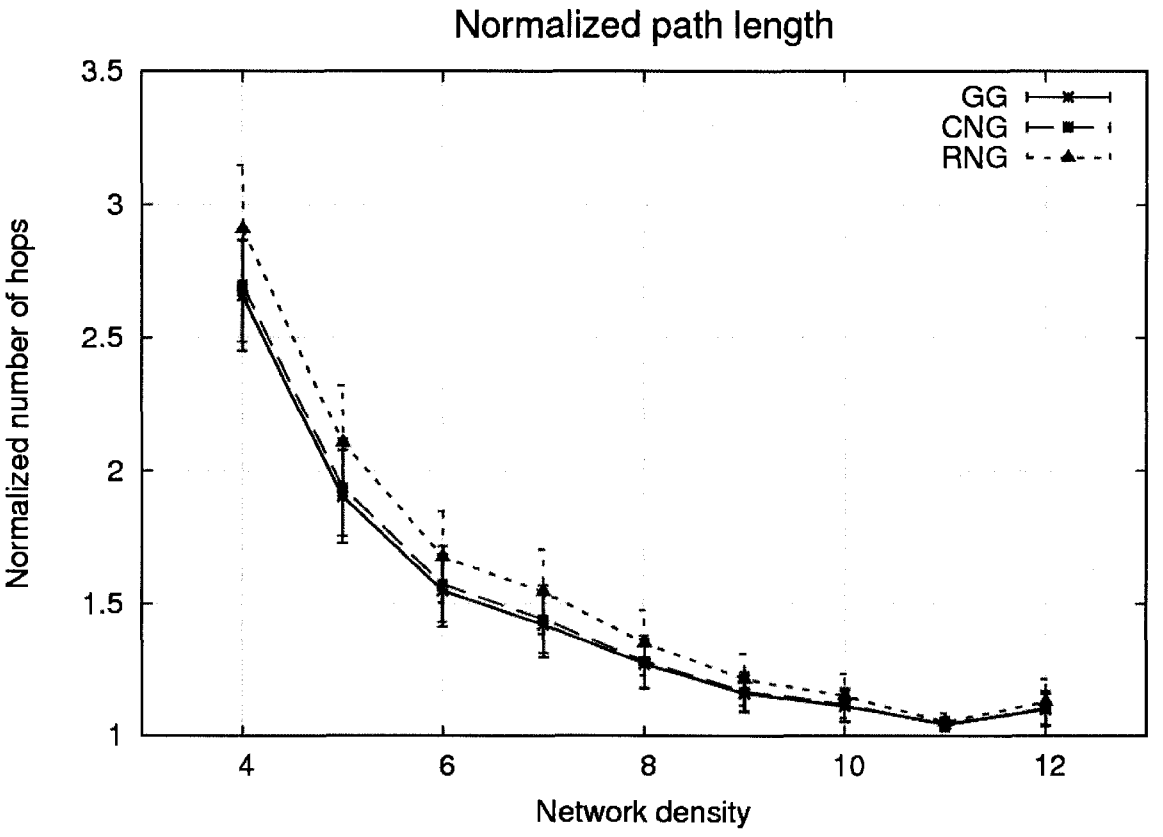


Fig. 7.11. Routing with BFP: Average normalized path length

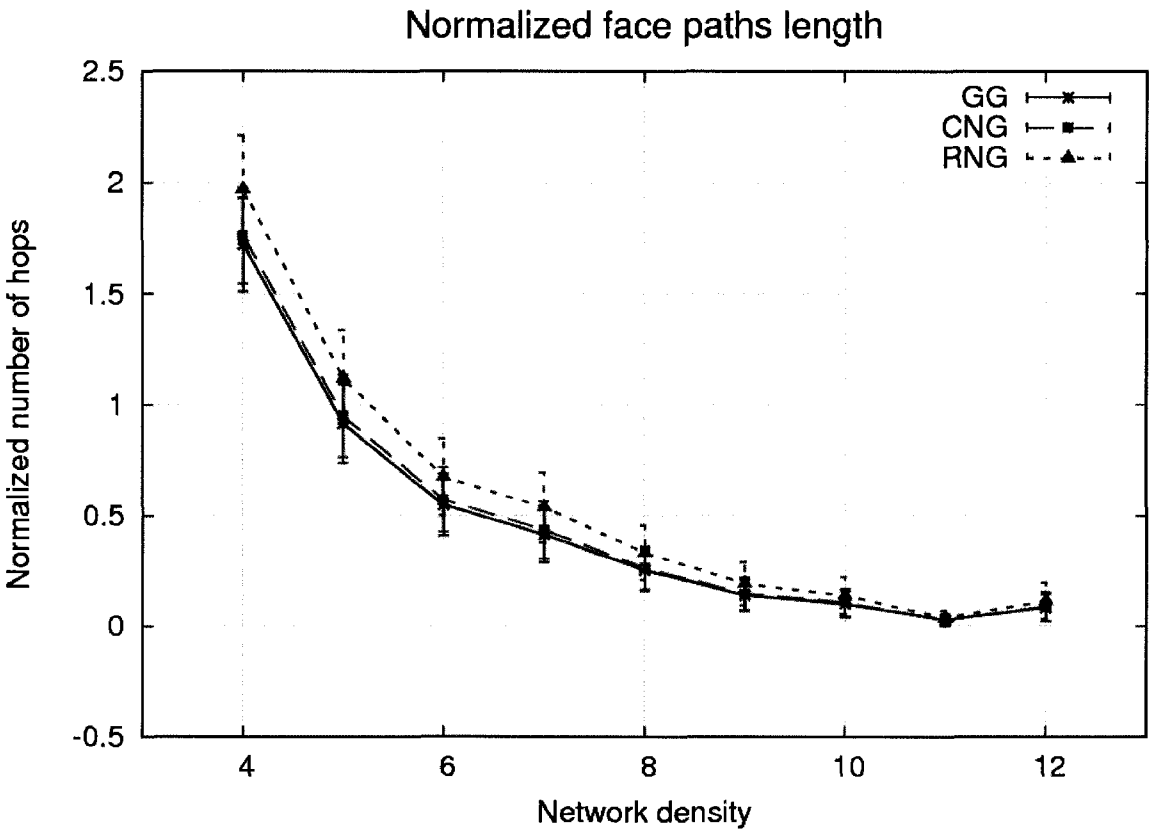


Fig. 7.12. Routing with BFP: Average normalized face path length

7.4 Angular Relaying

When using sweep curve instead of sweep line, we observe a reduction of the number of protests by more than a factor of 2 on average. This corresponds to our theoretical results showing a reduction of the expected protest area by the same factor. The reduction of protests leads to an overall message reduction of 11% on average. Note, that greedy routing is used for large parts of the routing path. We can also observe that BFP uses less protest messages than Angular Relaying. But that does not imply that BFP is more efficient, because some nodes that send a protest in Angular Relaying would send CTS when using BFP. This is reflected in the overall message complexity, where Angular Relaying seems more efficient. However, BFP constructs a complete local subgraph, whereas Angular Relaying determines only the next hop.

All the results are shown using the error bars diagrams using the 95% confidence interval that summarize simulations on 500 different random UDGs with 100 nodes each for every density value shown.

Our simulations show that the Angular relaying with sweep curve is advantageous in terms of number of protests as shown in Fig. 7.14. This leads to a better overall message complexity of the sweep curve variant (Fig. 7.15). Moreover, the maximum message complexity of this variant is always better that the sweep line (Fig. 7.16), although they produce the paths of the same length on average (Fig. 7.17) and require the same average recovery paths in order to guarantee delivery (Fig. 7.18)

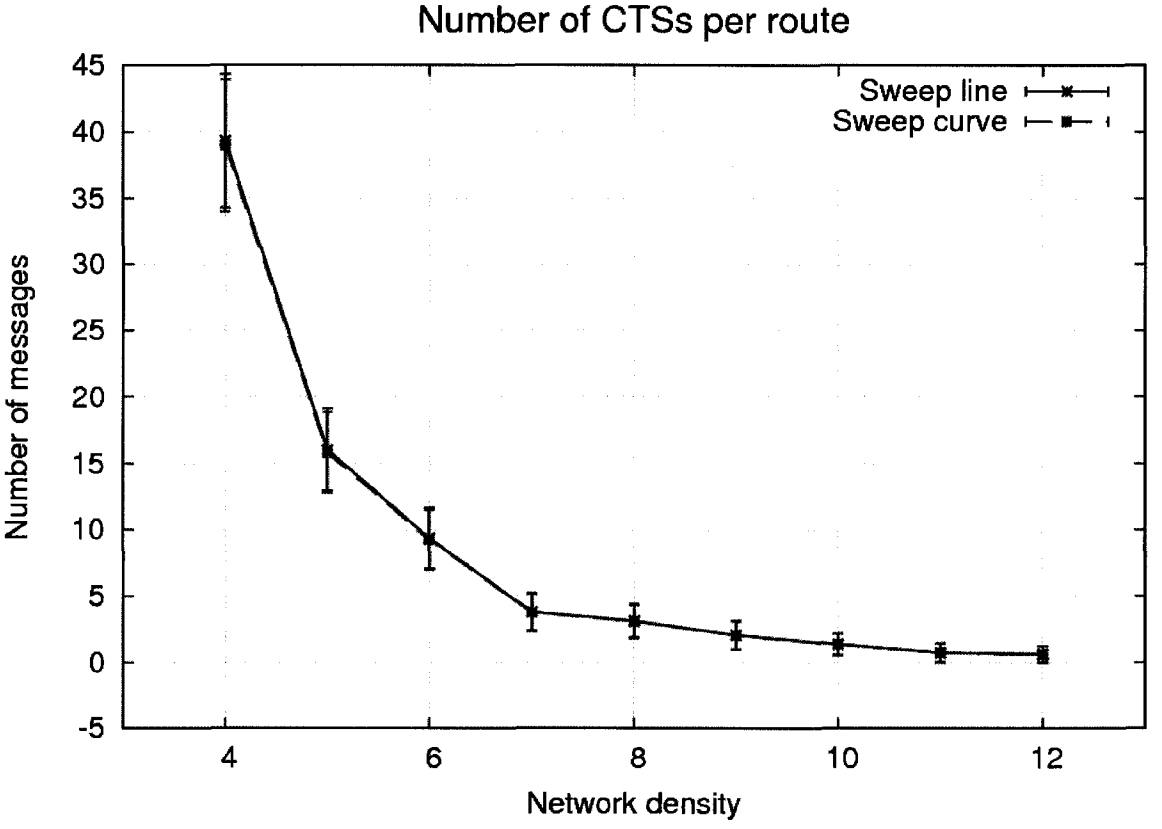


Fig. 7.13. Angular relaying: Average number of CTS messages per route

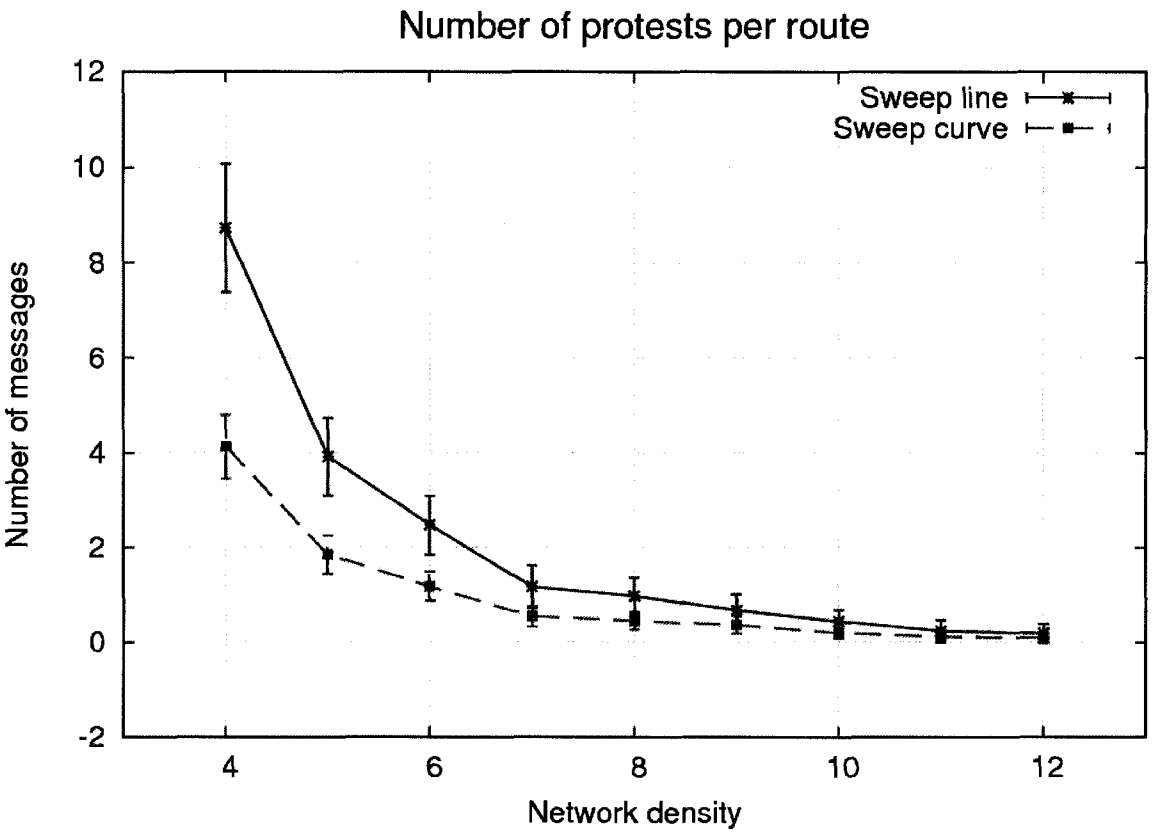


Fig. 7.14. Angular relaying: Average number of Protest messages per route

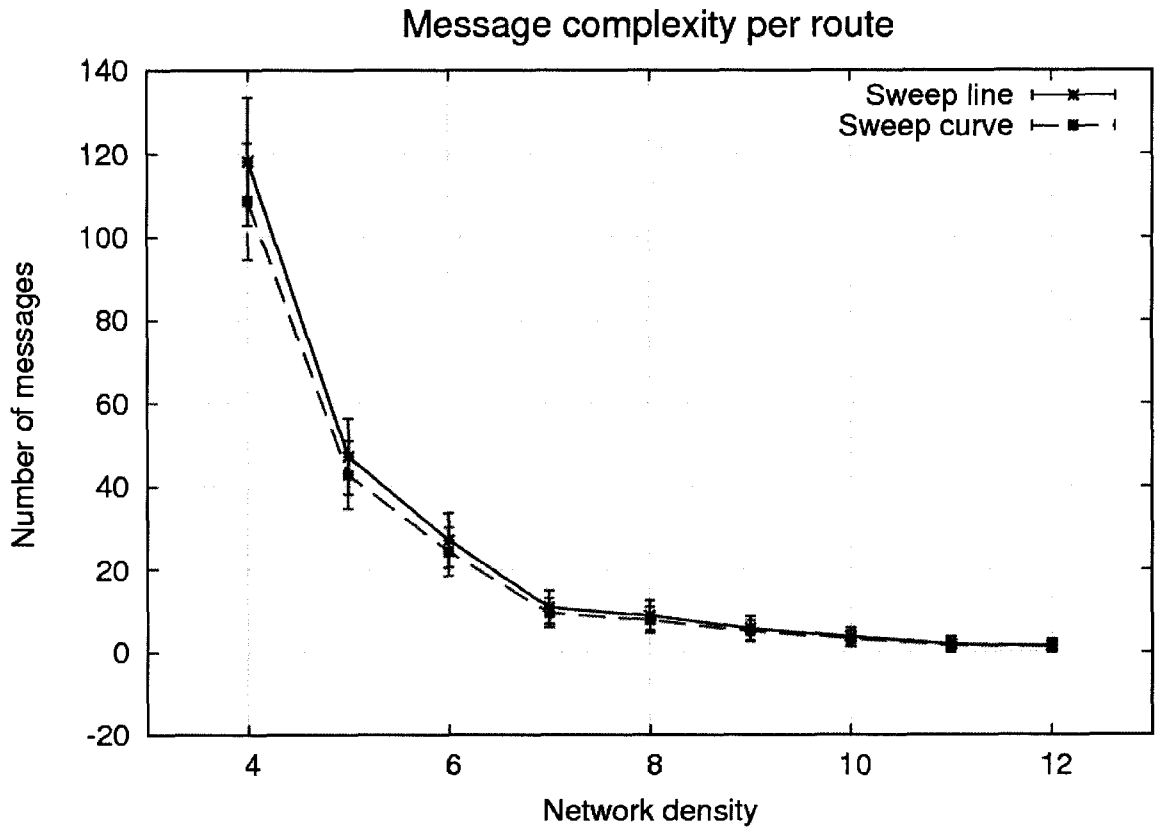


Fig. 7.15. Angular relaying: Average overall message complexity per route

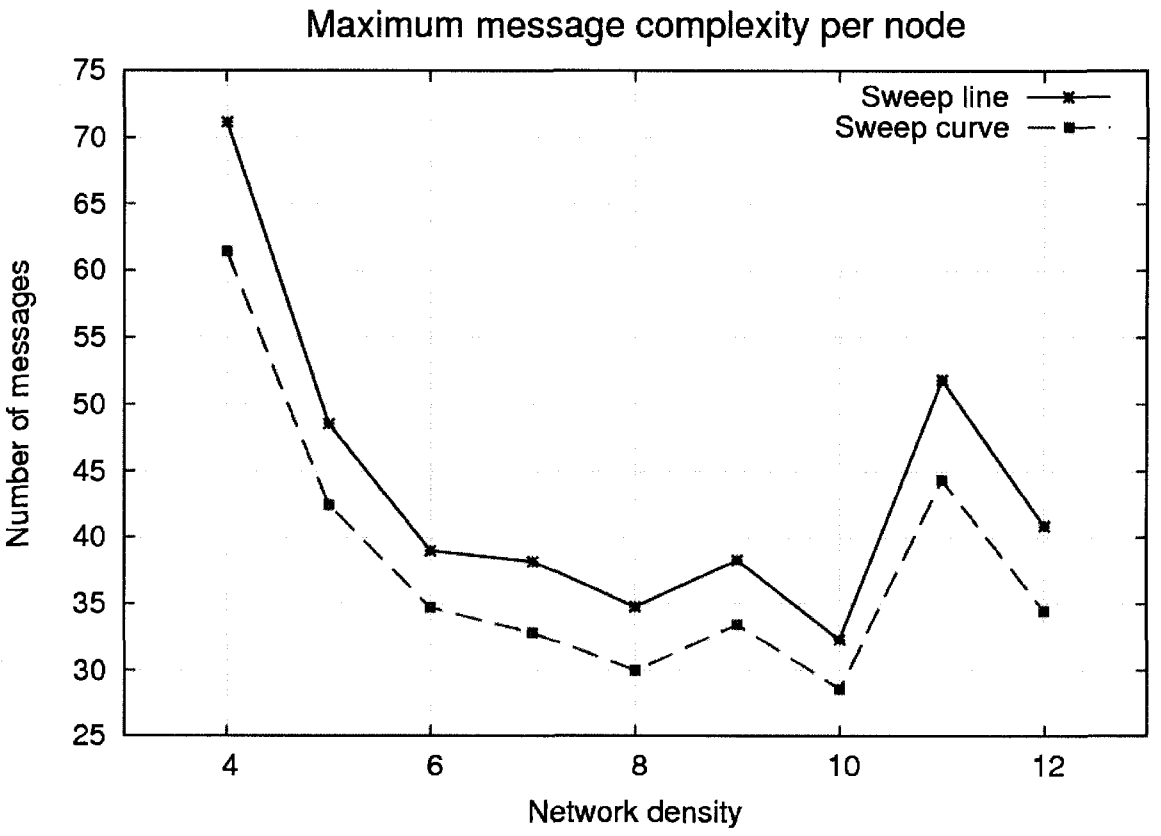


Fig. 7.16. Angular relaying: Maximum message complexity per node

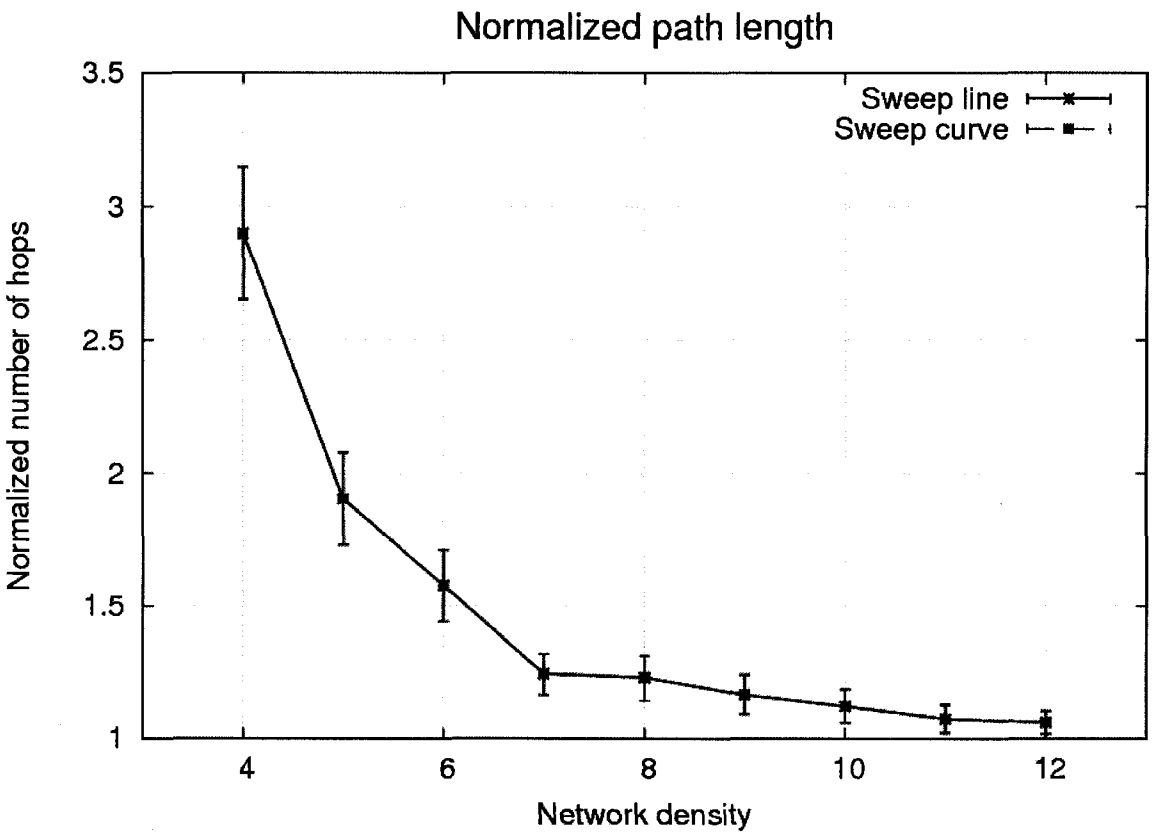


Fig. 7.17. Angular relaying: Average normalized path length

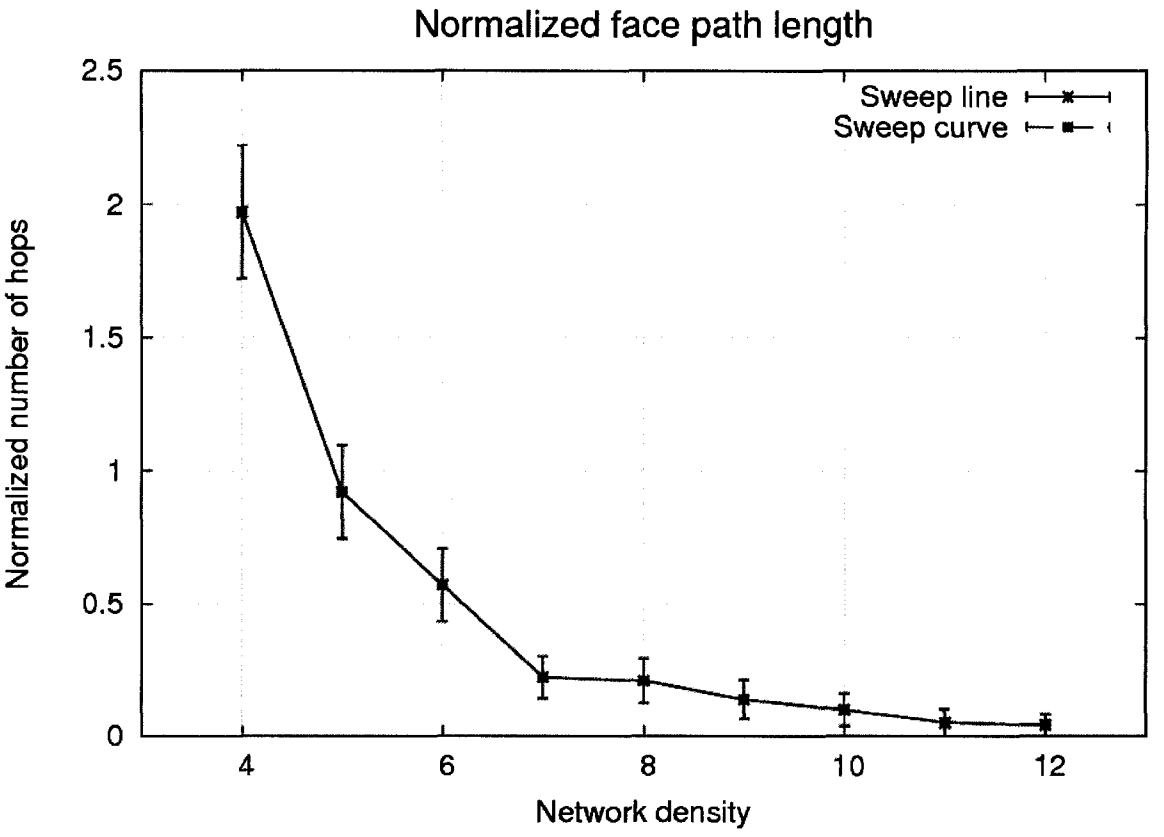


Fig. 7.18. Angular relaying: Average normalized face path length

7.5 Performance Comparison

Our simulations show that both BFP technique and Angular Relaying algorithm safely outperform BLR algorithm in terms of message complexity as shown on Fig. 7.19.

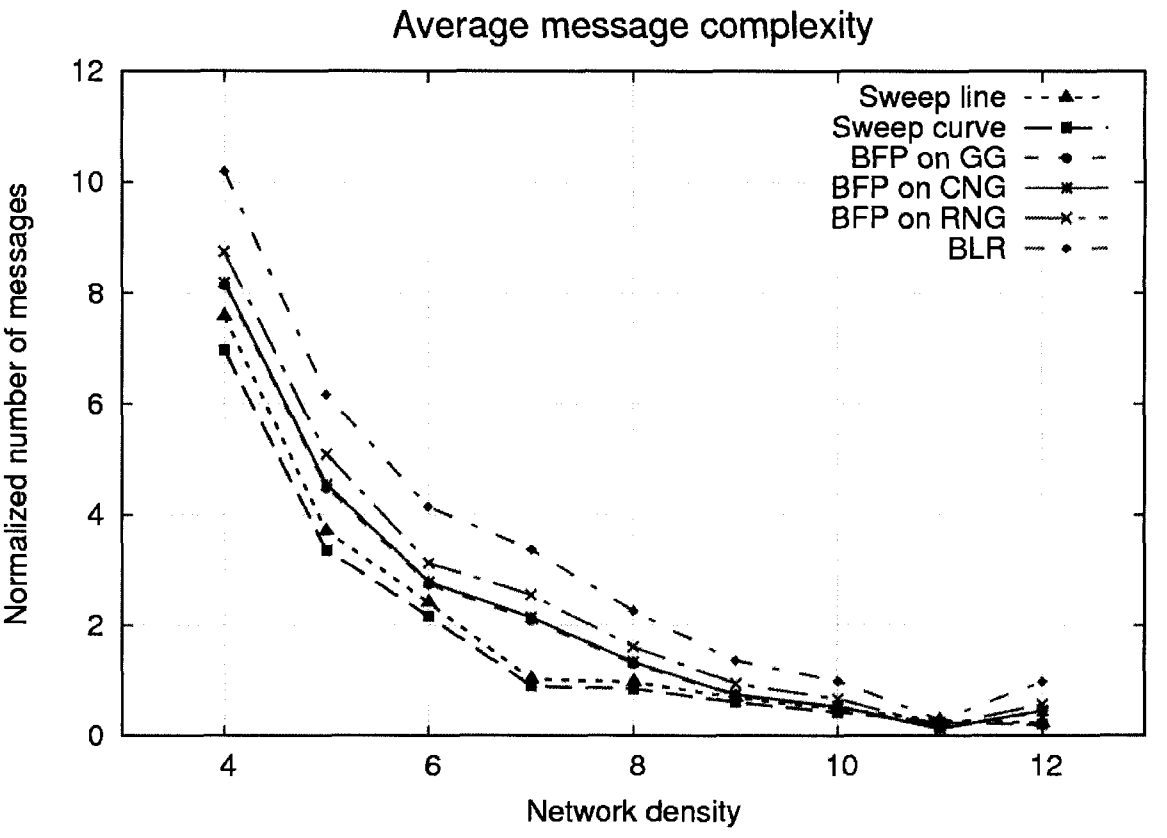


Fig. 7.19. Average normalized message complexity per node of Angular Relaying, BFP, and BLR algorithm

In terms of the average number of protests the performance of the Angular Relaying with sweep line is visibly worse than the performance of the sweep curve variant (Fig. 7.20). This is due to the fact that the sweep curve timer favors the nodes closer to the current forwarder and these nodes are more likely to be the Gabriel graph neighbors of the forwarder which leads to fewer protests against those nodes. The BFP produces even smaller number of protests than both of the Angular Relaying variants; however the overall message complexity of BFP is worse due to greater number of CTS messages (Fig. 7.19).

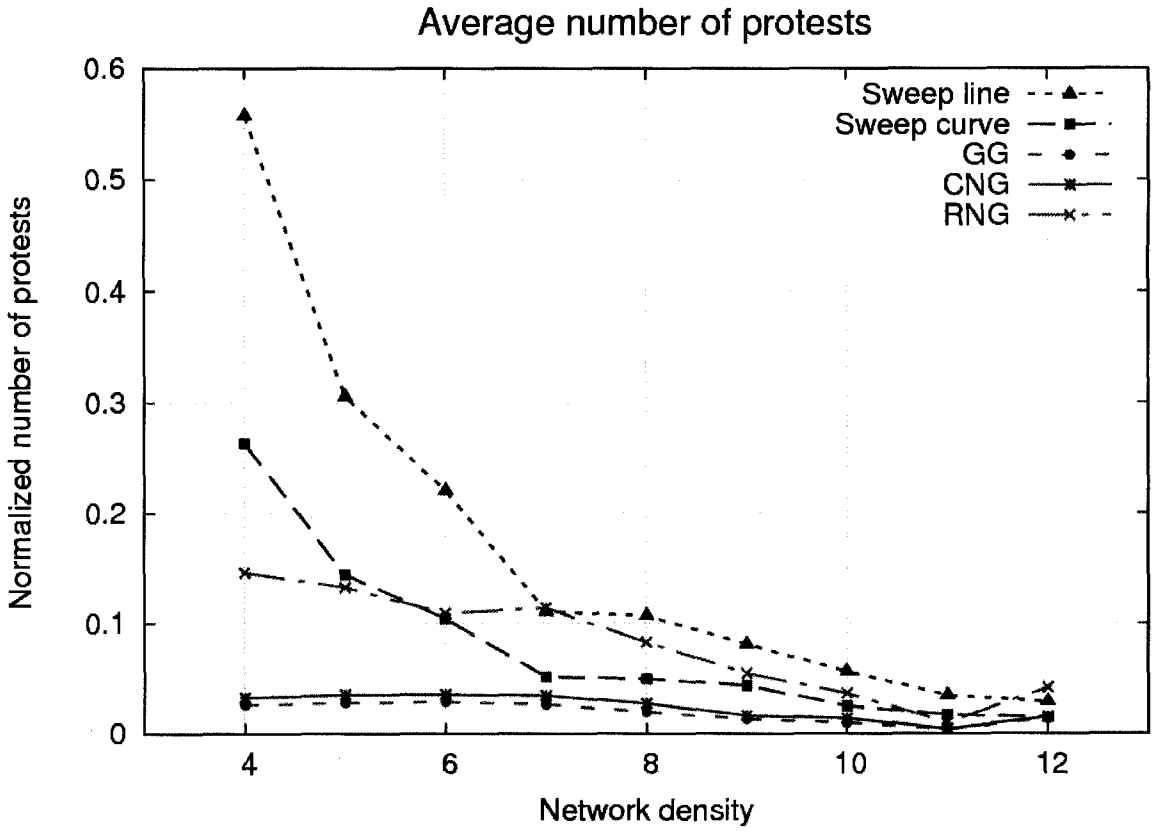


Fig. 7.20. Average normalized number of protests per node of Angular Relaying and BFP algorithm

7.6 Conventional GFG: Examples

The following is the example of two different paths generated by the GFG algorithm when RNG (red path) vs. CNG or GG (green path) is used in the recovery mode.

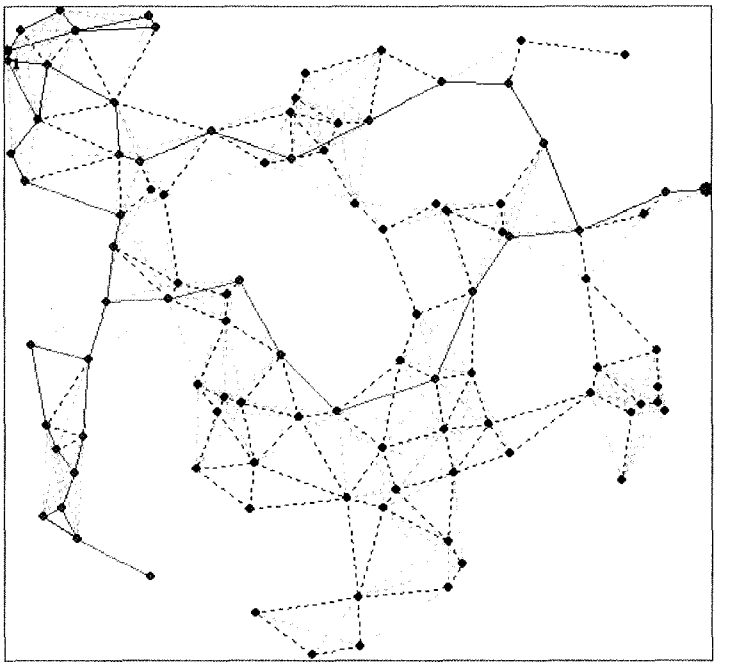


Fig. 7.21. Solid red (upper) path produced by GFG on RNG,
solid green path produced by GFG on both CNG and GG

The following is the example of two different paths generated by the GFG algorithm when RNG or CNG (red path) vs. GG (blue path) is used in the recovery mode.

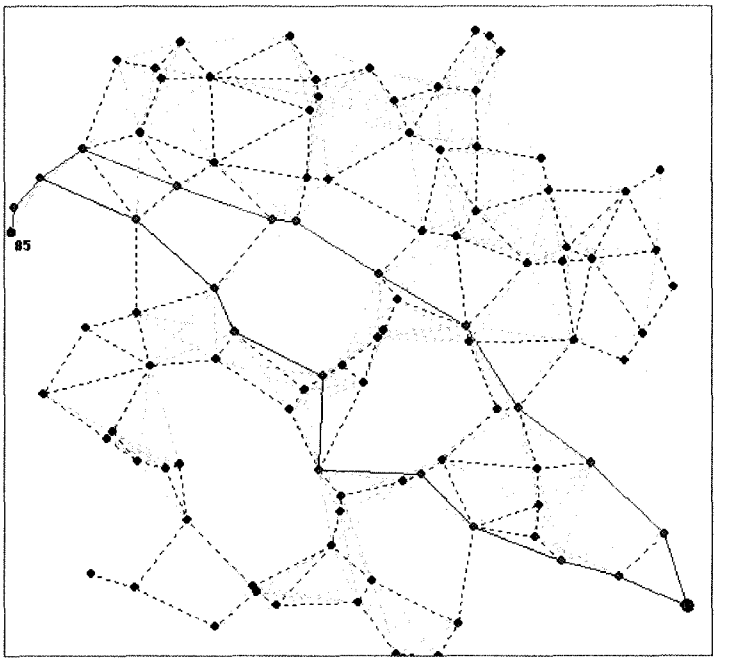


Fig. 7.22. Solid red (upper) path produced by GFG on both RNG and CNG,
solid blue path produced by GFG on GG

Chapter 8 Conclusion and Future Work

In this thesis the beaconless recovery problem has been addressed. As greedy routing fails in case of a local minimum, a recovery strategy is needed to provide guaranteed delivery. The preferred recovery method for conventional geographic routing algorithms is the face traversal on a planar subgraph, which is constructed from the neighborhood information. But for beaconless protocols, the full knowledge of the neighborhood is not available a priori. Instead, part of this knowledge has to be gained by exchanging messages, if it is not implicitly given by the location of the nodes.

We have identified two questions related to the beaconless recovery problem, the answer to which is the key to guaranteed delivery:

1) How to construct a local planar subgraph on the fly?

2) How to determine the next edge of a planar subgraph traversal?

We have presented two solutions for the beaconless recovery problem: Beaconless Forwarder Planarization algorithm which answers the first question, and Angular Relaying algorithm answering the second question.

We have also introduced a theoretical framework to analyze the message complexity of beaconless face routing algorithms. We could improve the message complexity by introducing a new planar subgraph construction (Circlunar Neighborhood graph) and new delay functions.

Our simulations of BFP on GG, CNG and RNG planar graphs have shown a gap between Gabriel graph and RNG for the number of protests and the overall message complexity. The inferior performance of RNG is due to the long detours caused by this planarization method. In BFP simulations the CNG reaches the good performance of the Gabriel graph, while guaranteeing a worst-case bound for the number of protests, which is not possible when using the Gabriel graph planarization.

In our simulations of the Angular Relaying algorithm, we have observed a reduction of the number of protests by more than a factor of 2 on average when using sweep curve instead of sweep line. This corresponds to our theoretical results showing a reduction of

the expected protest area by the same factor. The reduction of protests leads to an overall message reduction of 11% on average.

Further improvements could be achieved by remembering past transmissions in an RTS cache. Future research directions include extensions to handle discrete timeouts and collisions when using a realistic MAC layer.

References

[1] B. Blum, T. He, S. Son, and J. Stankovic, "IGF: A state-free robust communication protocol for wireless sensor networks," University of Virginia, USA, Tech. Rep. CS-2003-11, Apr. 2003.

[2] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, "On the spanning ratio of gabriel graphs and beta-skeletons," SIAM Journal on Discrete Mathematics, vol. 20(2), pp. 412–427, 2006.

[3] P. Bose, J. Gudmundsson, and P. Morin, "Ordered theta graphs," Computational Geometry, vol. 28, no. 1, pp. 11–18, May 2004.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in 3rd int. workshop on Discrete algorithms and methods for mobile computing and communications (DIALM '99). ACM Press, 1999, pp. 48–55.

[5] M. Chawla, N. Goel, K. Kalaichelvan, A. Nayak, and I. Stojmenovic, "Beaconless position-based routing with guaranteed delivery for wireless ad-hoc and sensor networks," in 1st IFIP Int. Conference on Ad-Hoc Networking. Springer, Aug. 2006, pp. 61–70.

[6] ——, "Beaconless position-based routing with guaranteed delivery for wireless ad hoc and sensor networks," ACTA AUTOMATICA SINICA, vol. 32, no. 6, pp. 846–855, Nov. 2006.

[7] D. Chen and P. K. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," IEEE Communications Surveys and Tutorials, 2007.

[8] R. J. Cimikowski, "Properties of some euclidean proximity graphs," Pattern Recognition Letters, vol. 13, no. 6, pp. 417–423, 1992.

[9] L. Devroye, "The expected size of some graphs in computational geometry," Computers & Mathematics with Applications, vol. 15, no. 1, pp. 53–64, 1988.

[10] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Univ. of Southern California, Tech. Rep. ISI/RR-87-180, Mar. 1987.

[11] H. Frey and I. Stojmenovic, "Geographic and energy aware routing in sensor networks," in Ivan Stojmenovic, editor, Handbook on Sensor Networks. Wiley, 2005.

[12] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks," in 12th Annual Int. Conference on Mobile Computing and Networking (MobiCom'06), 2006.

[13] H. Füßler, J. Widmer, M. Mauve, and H. Hartenstein, "A novel forwarding paradigm for position-based routing (with implicit addressing)," in IEEE 18th Annual Workshop on Computer Communications (CCW 2003), Oct. 2003, pp. 194–200.

[14] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," Systematic Zoology, vol. 18 (3), pp. 259–278, 1969.

[15] J. Gao, L. J. Guibas, J. E. Hershberger, L. Zhang, and A. Zhu, "Geometric spanner for routing in mobile networks," in 2nd Symposium on Mobile Ad Hoc Networking and Computing, Oct. 2001, pp. 45–55.

[16] S. Giordano and I. Stojmenovic, "Position based routing algorithms for ad hoc networks: A taxonomy," in Xiuzhen Cheng, Xiao Huang, and Ding-Zhu Du, editors, Ad Hoc Wireless Networking, pages 103–136. Kluwer, 2004.

[17] M. Heissenbüttel and T. Braun, "A novel position-based and beacon-less routing algorithm for mobile ad-hoc networks," in 3rd IEEE Workshop on Applications and Services in Wireless Networks, 2003, pp. 197–209.

[18] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, "BLR: Beacon-less routing algorithm for mobile ad-hoc networks," Computer Communications, vol. 27, no. 11, pp. 1076–1086, Jul. 2004.

[19] J. Hightower and G. Borriella, "Location systems for ubiquitous computing," IEEE Computer, 34(8):57–66, 2001.

[20] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," Proc. of the IEEE, vol. 80, pp. 1502–1517, 1992.

[21] H. Kalosha, A.Nayak, S. Rührup, I. Stojmenovic, "Select and Protest based Beaconless Georouting with Guaranteed Delivery in Wireless Sensor

Networks," accepted for publication in Proceedings of the 27th Conference on Computer Communications (IEEE INFOCOM 2008).

[22] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in Proceedings of the 6th ACM/IEEE Annual International Conference on Mobile Computing and Networking (MOBICOM-00), 2000.

[23] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in Proceedings of the 22nd ACM International Symposium on the Principles of Distributed Computing (PODC), pages 63–72, Boston, Massachusetts, USA, 2003.

[24] X.-Y. Li, G. Calinescu, and P.-J. Wan, "Distributed construction of planar spanner and routing for ad hoc wireless networks," in 21st Annual IEEE Conference on Computer Communications (INFOCOM), 2002.

[25] X.-Y. Li, "Approximate MST for UDG locally," in 9th Annual Int. Conference on Computing and Combinatorics, 2003, pp. 364–373.

[26] X.-Y. Li, I. Stojmenovic, and Y. Wang, "Partial delaunay triangulation and degree limited localized bluetooth scatternet formation," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 4, pp. 350–361, 2004.

[27] M. Narasawa, M. Ono, and H. Higaki, "NB-FACE: No-beacon face adhoc routing protocol for reduction of location acquisition overhead," in 7th Int. Conf. on Mobile Data Management (MDM'06), 2006, p. 102.

[28] F. A. Onat and I. Stojmenovic, "Generating Random Graphs for Wireless Actuator Networks," 8th IEEE Symposium on a World of Wireless, Mobile and Multimedia Networks WoWMoM, Helsinki, June 18-21, 2007.

[29] I. Stojmenovic, "Location updates for efficient routing in ad hoc networks," in Ivan Stojmenovic, editor, Handbook of Wireless Networks and Mobile Computing, chapter 21, pages 451–471. Wiley, 2002.

[30] I. Stojmenovic and X. Lin, "Loop-free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks," IEEE Trans. Parallel Dist. Sys., vol. 12, no. 10, 2001, pp. 1023–32.

[31] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," IEEE Transactions on Parallel and Distributed Systems, 12(11):1122–1133, November 2001.

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

[33] M. Witt and V. Turau, "BGR: Blind geographic routing for sensor networks," in 3rd Workshop on Intelligent Solutions in Embedded Systems (WISES'05), 2005, pp. 51–61.

[34] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "PSGR: Priority-based stateless geo-routing in wireless sensor networks," in 2nd IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems (MASS'05), 2005.

[35] M. Zorzi, "A new contention-based MAC protocol for geographic forwarding in ad hoc and sensor networks," in IEEE Int. Conference on Communications (ICC 2004), 2004, pp. 3481–3485.