DISS. ETH NO. 16025

Networking Unleashed:

Geographic Routing and Topology Control in Ad Hoc and Sensor Networks

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Sciences

presented by

AARON ZOLLINGER

Dipl. Inf.-Ing., ETH Zürich

born 24.04.1975

citizen of

Zürich ZH and Regensdorf ZH

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, examiner
Prof. Dr. Matthias Grossglauser, co-examiner
Charles E. Perkins, co-examiner

2005

Abstract

Ad hoc and sensor networks consist of autonomous devices communicating via radio equipment. Common scenarios for ad hoc networks include survivable, efficient, dynamic communication networks for emergency and rescue operations, disaster relief efforts, and similar tasks where typically no communication infrastructure is present prior to the deployment of the ad hoc network. In sensor networks, nodes are additionally equipped with sensors, performing the task of sensing a certain physical value, such as temperature, humidity, brightness, or motion, and periodically reporting the sensed data to a designated sink node for monitoring purposes.

Since ad hoc and sensor network nodes are generally assumed to be autonomous and operate for a considerable period of time—in case of sensor networks up to several years—, energy conservation is one of the central issues in this research context. On the other hand, many scenarios assume a high degree of dynamics, particularly based on node mobility.

This dissertation discusses two major problem fields in the context of ad hoc and sensor networks. In particular, geographic routing—a local type of routing inherently well suited for dynamic ad hoc networks—is studied with respect to both worst-case and average-case networks. Second, topology control based on transmission power reduction puts the focus on energy conservation as a consequence of restricted interference among the network nodes.

Zusammenfassung

Ad-Hoc- und Sensornetze bestehen aus unabhängigen über Funk kommunizierenden Geräten. Häufig genannte Szenarien für Ad-Hoc-Netzwerke beschreiben ihren Einsatz als robuste, sparsame und dynamische Kommunikationsnetze für Notfall- und Rettungseinsätze, Katastrophenhilfe und ähnliche Aufgaben, wo vor dem Einsatz des Ad-Hoc-Netzwerkes keine Kommunikationsinfrastruktur vorhanden ist. In Sensornetzen sind die Netzwerkknoten zusätzlich mit Sensoren ausgestattet, die bestimmte physikalische Grössen – wie beispielsweise Temperatur, Feuchtigkeit, Helligkeit oder Bewegung – messen; diese Daten werden periodisch an einen vorbezeichneten Sammelknoten gesendet und zum Zwecke der Beobachtung oder der Kontrolle ausgewertet.

Da üblicherweise angenommen wird, dass Knoten in Ad-Hoc- und Sensornetzen unabhängig von externen Energiequellen sind und trotzdem über eine beträchtliche Zeitspanne betrieben werden sollen – in Sensornetzen bis zu mehreren Jahren –, ist Energieeffizienz ein Hauptpunkt dieser Forschungsrichtung. Andererseits beinhalten viele Szenarien hohe Netzwerkdynamik, insbesondere aufgrund mobiler Knoten.

Diese Dissertation befasst sich mit zwei zentralen Problemfeldern im Bereich von Ad-Hoc- und Sensornetzen. Im Einzelnen wird das Verhalten von geografischem Routing – einer auf lokal begrenzter Information fussenden Art von Nachrichtenvermittlung, die speziell gut für den Einsatz in dynamischen Ad-Hoc-Netzwerken geeignet ist – in schlechtest möglichen und in durchschnittlich auftretenden Netzwerken untersucht. In einem zweiten Teil der Arbeit wird der Schwerpunkt auf Topologiekontrolle im Zusammenhang mit gezielt beschränkter Funksendeleistung gelegt; Energieeffizienz ist hier eine Folge von verringerter Signalstörung, oder „Interferenz“, zwischen den Netzwerkknoten.

Acknowledgements

First of all I would like to thank my advisor Roger Wattenhofer for guiding me through the years of my PhD studies. It is true, you always took your time to discuss any technical—or other—issues with me. It was also a very interesting and enriching experience to witness how (y)our Distributed Computing Group grew over the years. I hope I am worth being your first "doctoral son".

I would also like to express my gratitude to my co-examiners Matthias Grossglauser and Charles Perkins who both helped me improve my thesis with their profound and inspiring comments.

Among the most important addressees of my gratefulness are of course all members of the Distributed Computing Group. In particular, I would like to thank Fabian "Let's solve a problem" Kuhn for being my patient office mate for all those years, for being my daily company, and for offering his skills to be my hitherto most important co-author (with the exception of Roger, of course); Keno "I always picture Switzerland upside-down" Albrecht for offering me a German point of view on all matters discussed—by the way, Keno is the only German I know of who can enumerate all Swiss cantons with their capitals—and for relieving me from spamania; Ruedi "I feel good!" Arnold—the only real didact in our group—for giving me a sportive flavor of pedagogy and for being a challenging table tennis opponent; Regina "Квантовая физика—единственная истинная наука" O'Dell for pointing out a woman's perspective on all things there are and for making fun of me on every possible and impossible occasion; Pascal "We're on a highway to hell" von Rickenbach for reminding us time and again where we all are, for his almost ubiquitous grim sense of humor, and last but not least for guiding me safely through all blizzards to my doctoral exam; Thomas "All I know, I know it from you" Moscibroda for his mastery of middle names, his daily enrichment of my work hours with original word creations and for behaving with due reverence; Nicolas "I will solve any problem provided that it's formulated correctly" Burri for being my didactic conscience and for showing me a new perspective on human love for ruminants; Stefan "My favorite place was the food court" Schmid for being a pleasant companion in exploring unsought highways, hoping that you will always be able to distinguish between peer-to-peer and beer-to-beer; Andi "Winning to nil would be a good idea" Wetzel for being a huge help above all during the last months with my frequent departures and last-minute hardware and software requests; Fränzi "in silico" Humair, my second unofficial office mate, for allowing me to witness the creation of a Master thesis in biology.

Incidentally, you have all been very nice table soccer opponents; above all I appreciate that you would let me head the ranking most of the time. You can now stop playing therapeutically with me...

Of course also the work contributed by the many students I had the pleasure to assist in their theses is invaluable. I am grateful to the Diploma and Master students Martin Burkhart, Pascal von Rickenbach (in this role), and Martin Fussen, as well as to the "semester thesis students" Björn Glaus, Marc Schiely,

Clemens Schroedter, Christian Gegenschatz, Nicolas Burri (also in this role), André Bayer, Yves Weber, Thomas Locher, Frank Lyner, and Philip Frey.

Then I would like to thank Joachim Giesen for actually pointing me towards Roger's newly founded research group and for helping me take my first steps in the world of research. I would also like to apologize for forgetting to invite you to my doctoral exam.

And last but not least I would like to express my utmost gratitude to my parents Sara and Kên and the whole of my family. You all know that I would not have achieved this goal without your support and warmth in the past, the present, and the future.

Contents

Chapter 1

Introduction: Of Theory and Practice

In theory, there is no difference between theory and practice; in practice, there is.

There is nothing more practical than a good theory.

One manifestation of the currently observed and continuing miniaturization of electronics in general and wireless communication technology in particular is mobile ad hoc networks. Ad hoc networks are formed by mobile devices consisting of, among other components, a processor, some memory, a radio communication unit, and a power source, due to physical constraints commonly a weak battery or a small solar cell.

Typically, wireless ad hoc networks are intended to be employed where no communication infrastructure is present before the deployment of the ad hoc network or where reliance on previously present infrastructure is not desired or not possible. Common scenarios for ad hoc networks include communication among rescue teams, police squads, or during fire fighting or other disaster relief actions. Another often mentioned scenario involves cars forming an ad hoc network for professional, entertainment, or informational purposes. Car-mounted radio broadcast warning systems automatically alerting approaching automobiles of accidents or other unexpected traffic events are frequently envisioned. Also for meetings or conferences ad hoc networks may have their value. Furthermore, ad hoc networks may find their application for security and—inevitably—for military purposes.[1]

Sensor networks can be considered a specialization of ad hoc networks in which nodes are equipped with sensors measuring certain physical values, such as humidity, brightness, temperature, acceleration, or vibration. Usually, the

[1]This raises the interesting question whether there actually exists any technology that cannot be applied for military purposes.

sensor nodes are designed to report measured information to a data sink node. Among the most common scenarios for sensor networks are environmental monitoring tasks, for instance to warn of imminent natural disasters or for the purpose of biological or other scientific observations. Typically, sensor networks will be deployed in areas difficult to access or, more generally, where human presence or stationary monitoring infrastructure is undesired or impossible.

Ad hoc and sensor networks are "unleashed" in two respects: First, network nodes communicate via radio technology. Second, nodes are independent of external power sources. As a consequence of this autonomy, ad hoc network nodes are often assumed to be mobile. Together with the fact that wireless links are inherently less stable and reliable than wired connections, node mobility leads to potentially highly dynamic networks. Another implication of the autonomy of network nodes is that energy is one of the most critical resources in ad hoc networks.

In this dissertation we will describe two approaches addressing the key issues of dynamics and energy consumption in ad hoc and sensor networks. In a first part of the dissertation, geographic routing will be shown to be a type of routing particularly well suited for application in dynamic networks, mainly due to its property of being based on completely local operations. A second part of the dissertation will focus on energy consumption; in particular, interference as one of the main sources for avoidable energy consumption forms the central aspect of various topology control techniques.

It is almost a truism that a purely theoretical analysis of a system runs the risk of producing a method or technique that may prove good in theory but turns out to be impracticable or at least less favorable in practice. On the other hand, mere practical evaluation of a proposed method does not necessarily lead to a satisfying assessment of the quality of the entity under examination. In this dissertation we attempt to narrow the all too often seemingly insurmountable gap between theory and practice. We maintain an analytical algorithmic perspective in that we propose algorithms with provable guarantees and properties. We however try to go beyond pure theory by studying average-case behavior of the proposed algorithms or by avoiding unrealistic assumptions. In the first part of the dissertation, for instance, we will present a geographic routing algorithm that is proved by means of theoretical analysis to be asymptotically optimal with respect to cost in worst-case networks; at the same time, this algorithm will however be shown to be, to the best of our knowledge, the fastest and cheapest known geographic routing algorithm also in average-case networks. A second example is the lightweight topology control algorithm presented at the beginning of the second part. This algorithm is intended to unify theory and practice in that it features provable properties while being independent of unrealistic assumptions and at the same time simple enough to be implemented in practical networks. The remainder of the second part is dedicated to the attempt of providing the problem field of interference in wireless ad hoc networks—which can be expected to be one of the major issues in practical networks—with a solid theoretical underpinning.

Part I

Geographic Routing

Chapter 2

Geographic Routing: An Introduction

Philosophy, n. A route of many roads leading from nowhere to nothing.
Ambrose Bierce (1842–1914)

Routing in a communication network is the process of forwarding a message from a source host to a destination host via intermediate nodes. In wired networks, routing is commonly a task performed by routers, special fail-safe network hosts particularly designed for the purpose of forwarding messages with high performance. In ideal wireless ad hoc networks, in contrast, every network node may act as a router, as a relay node forwarding a message on its way from its source node to its destination node. This process is particularly important in ad hoc networks, as network nodes are assumed to have restricted power resources and therefore try to transmit messages at low transmission power, leading to the effect that the destination of a message can typically not be reached directly from the source. The importance of this task also becomes manifest in the popular term *multihop routing*, expressing the essential role of network nodes as relay stations.

In wired networks, routing almost always takes place in relatively stable conditions; at least the main neighborhood topology remains identical over weeks, months, or even years. The main focus of routing in wired networks is on high-performance forwarding of messages; reaction latency in the face of network topology changes, caused by failing hosts or connections, is generally of secondary importance. Considering the stability of wired networks, prompt reaction to topology changes or rapid propagation of according information is often not required, as such events are relatively rare.

Wireless ad hoc networks are of a fundamentally different character: To begin with, wireless connections are by nature significantly less stable than wired connections. Effects influencing the propagation of radio signals, such as shield-

ing, reflection, scattering, and interference, inevitably require routing systems in ad hoc networks to be able to cope with comparatively low link communication reliability. More importantly, many scenarios for ad hoc networks assume that nodes are potentially mobile. These two factors, above all in high node mobility, cause ad hoc networks to be inherently more dynamic than wired networks. Traditional routing protocols designed for wired networks therefore generally fail to satisfy the requirements of wireless ad hoc networks.

A considerable number of routing protocols specifically devised for operation in ad hoc networks have consequently been invented. These protocols are usually classified into two groups: *proactive* and *reactive* routing protocols. *Proactive* routing protocols resemble protocols for wired networks in that they collect routing information ahead of time. A request for a message to be routed can be serviced without any further preparative actions. As every node keeps a table specifying how to forward a message, information on topology changes is propagated whenever they occur. Similar to routing protocols in wired networks, proactive routing protocols are efficient only if links are stable and node mobility is low compared to the rate of communication traffic. Already if node mobility reaches a reasonable degree, the routing overhead incurred by table update messages can become unacceptably high. Another question is whether lightweight ad hoc network nodes with scarce resources can be expected to maintain routing tables potentially for all possible destinations in the network. *Reactive* routing protocols, on the other hand, try to delay any preparatory actions as long as possible. Routing occurs on demand, only. In principle, a node wishing to send a message has to flood the network in order to find the destination. Although there are many tricks to restrict flooding or to cache information overheard by nodes, flooding can consume a considerable portion of the network bandwidth. Attempting to combine the advantages of both concepts, proposals have also been made to incorporate both approaches in hybrid protocols, adapting to current network conditions.

Most of these routing protocols have been described and studied from a system-centric point of view. Simulation appears to be the preferred method of assessment. It appears, however, that a global evaluation of protocols is difficult. Ad hoc networks have many parameters, such as transmission power, signal attenuation, interference, physical obstacles, node density and distribution, degree and type of node mobility, just to mention a few; therefore simulation cannot cover all the degrees of freedom. For a given set of parameters, certain protocols appear superior to others; for other parameters, the ranking may be reversed. One possible answer to this problem may be found in trying to rigorously analyze the efficiency of proposed protocols and algorithms. However, analyzing the complexity of ad hoc routing algorithms appears to be not only intricate, but virtually impossible. Accordingly, only few attempts have been made to analyze ad hoc routing in a general setting from an algorithmic perspective.

One specific type of ad hoc routing, in contrast, appears to be more easily accessible to algorithmic analysis: geographic routing. Geographic routing, sometimes also called directional, geometric, location-based, or position-based

routing, is based on two principal assumptions. First, it is assumed that every node knows its own and its network neighbors' positions. Second, the source of a message is assumed to be informed about the position of the destination. The former assumption becomes more and more realistic with the advent of inexpensive and miniaturized positioning systems. It is also conceivable that position information could be attained by local computation and message exchange with stationary devices. In order to come up to the latter assumption, that is to provide the source of a message with the destination position, several so-called location services have been proposed. For some scenarios it can also be sufficient to reach *any* destination currently located in a given area, sometimes called "geocasting". These are only briefly summarized explanations why the two basic assumptions of geographic routing are reasonable. This issue will be discussed later in more depth.

Geographic routing is particularly interesting, as it operates without any routing tables whatsoever. Furthermore, once the position of the destination is known, all operations are strictly local, that is, every node is required to keep track only of its direct neighbors. These two factors—absence of necessity to keep routing tables up to date and independence of remotely occurring topology changes—are among the foremost reasons why geographic routing is exceptionally suitable for operation in ad hoc networks. Furthermore, in a sense, geographic routing can be considered a lean version of source routing appropriate for dynamic networks: While in source routing the complete hop-by-hop route to be followed by the message is specified by the source, in geographic routing the source simply addresses the message with the position of the destination. As the destination can generally be expected to move slowly compared to the frequency of topology changes between the source and the destination, it makes sense to keep track of the position of the destination instead of maintaining network topology information up to date; if the destination does not move too fast, the message is delivered regardless of possible topology changes among intermediate nodes. Finally, from a less technical perspective, it can be hoped that by studying geographic routing it is possible to gain insights on routing in ad hoc networks in general, without availability of position information.

We will start our analysis of geographic routing by describing a simple greedy routing approach in Chapter 4. The main drawback of this approach is that it cannot guarantee to always reach the destination. Geographic routing algorithms that, in contrast, always reach the destination, are based on faces, contiguous regions separated by the edges of planar network subgraphs. It may however happen that these algorithms take $\Omega(n)$ steps before arriving at the destination, where n is the number of network nodes. In other words, they basically do not perform better than an algorithm visiting every node in the network. In Chapter 5 we will describe the concept of face routing and describe algorithms that not only always find the destination, but are also guaranteed to do so with cost at most $\mathrm{O}(c^2)$, where c is the cost of a shortest path connecting the source and the destination. The next chapter will show that, given an instance of a class of lower bound graphs, no geographic routing algorithm will be able to

perform better; in this sense, the presented face routing algorithms are asymptotically optimal in worst-case networks. Despite their asymptotic optimality, these algorithms are relatively inflexible in that they follow the boundaries of faces also in dense average-case networks where greedy routing would reach the destination much faster. Chapter 7 will describe how greedy routing and face routing can be combined, resulting in the GOAFR and GOAFR$^+$ algorithms, which preserve the worst-case guarantees of their face routing components. In addition, the comprehensive simulations presented in Chapter 8 will show that the GOAFR$^+$ algorithm is—to the best of our knowledge—the currently most efficient geographic routing algorithm also in average-case networks. GOAFR$^+$ particularly outperforms other routing algorithms in a critical node density range, where the network is just about to become connected and which forms a challenge to any routing algorithm, also non-geographic routing algorithms.

The results presented up to Chapter 8 are based on the $\Omega(1)$-model, the assumption that the distance between any pair of nodes cannot be smaller than a constant value. In Chapter 9 we will show that an equivalent property can be achieved by computation of a subgraph of the network acting as a routing backbone. More exactly, this chapter will show that the set of possible cost metrics falls into two classes; the equivalence of the routing backbone technique with the $\Omega(1)$-model holds for one class of cost metrics, whereas it will be shown that for metrics in the other class, networks are constructible in which no geographic routing algorithm can reach the destination with cost comparable to the cost of a shortest path between the source and the destination.

If the analytical results discussed so far are based on the unit disk graph model, where transmission ranges are modeled as disks with radii of one unit each, centered at the corresponding node, Chapter 10 will present and analyze a model that goes beyond unit disk graphs.

Two less technical chapters will conclude the first part of the dissertation: Chapter 11 will discuss and give reasons for the basic assumptions of geographic routing. Although our analysis assumes that routing occurs significantly faster than node mobility, graph dynamics—caused by fluctuating edges or moving nodes—is one of the most important issues in ad hoc networks. Therefore a number of issues and approaches in the context of mobility models will be addressed in Chapter 12.

Related Work

As mentioned earlier, routing protocols for ad hoc networks can be classified as proactive and reactive protocols. Proactive protocols, such as DSDV [88], TBRPF [84], and OLSR [25], distribute routing information ahead of time in order to be able to react immediately whenever a message needs to be forwarded. On the other hand, reactive protocols, such as AODV [89], DSR [55], or TORA [85] do not try to anticipate communication and initiate route discovery as late as possible, as a reaction to a message requested to be routed.

As the performance and incurred routing overhead of such protocols highly depends on the type and extent of network mobility, also hybrid protocols, such as [53, 83, 95] have been proposed. Further reviews of routing algorithms in mobile ad hoc networks in general can be found in [15] and [98]. Most of these protocols have been described and studied from a system perspective; performance and efficiency assessment was commonly carried out by means of simulation. To date, only few attempts have been made to analyze routing in ad hoc networks in a general setting from an analytical algorithmic perspective [17, 39, 92].

The early proposals of geographic routing—suggested over a decade ago—were of purely greedy nature: At each intermediate network node the message to be routed is forwarded to the neighbor closest to the destination [36, 47, 101]. This can however fail if the message reaches a local minimum with respect to the distance to the destination, that is a node without any "better" neighbors. Also a "least deviation angle" approach (*Compass Routing* in [61]) cannot guarantee message delivery in all cases.

The first geographic routing algorithm that does guarantee delivery was *Face Routing* introduced in [61] (called *Compass Routing II* there). *Face Routing* walks along faces of planar graphs and proceeds along the line connecting the source and the destination. Besides guaranteeing to reach the destination, it does so with $O(n)$ messages, where n is the number of network nodes. However, this is unsatisfactory, since also a simple flooding algorithm will reach the destination with $O(n)$ messages. Additionally it would be desirable to see the algorithm cost depend on the distance between the source and the destination.

There have been later suggestions for algorithms with guaranteed message delivery [14, 28]; at least in the worst case, however, none of them outperforms original Face Routing. Yet other geographic routing algorithms have been shown to reach the destination on special planar graphs without any runtime guarantees [12]. [13] proposed an algorithm competitive with the shortest path between source and destination on Delaunay triangulations; this is however not applicable to ad hoc networks, as Delaunay triangulations may contain arbitrarily long edges, whereas transmission ranges in ad hoc networks are limited. Accordingly, [41] proposed local approximation of the Delaunay Graph, however without improving performance bounds for routing. A more detailed overview of geographic routing can be found in [103].

In [66] we proposed *Adaptive Face Routing* AFR. The execution cost of this algorithm—basically enhancing Face Routing by the employment of an ellipse restricting the searchable area—is bounded by the cost of the optimal route. In particular, the cost of AFR is not greater than the squared cost of the optimal route. We also showed that this is the worst-case optimal result any geographic routing algorithm can achieve.

Face Routing and also AFR are not applicable for practical purposes due to their strict employment of face traversal. There have been proposals for practical purposes to combine greedy routing with face routing [14, 28, 57], however without competitive worst-case guarantees. In [68] we introduced the GOAFR algorithm discussed later in Chapter 7; to the best of our knowledge,

this was the first algorithm to combine greedy and face routing in a worst-case optimal way; the GOAFR$^+$ algorithm [65] remains asymptotically worst-case optimal while improving GOAFR's average-case efficiency by employing a counter technique for falling back as soon as possible from face to greedy routing.

The results in the first part of this dissertation partly rely on the $\Omega(1)$-model, the assumption that the distance between any pair of network nodes is at least a (possibly small) constant. In Chapter 9 we will show that equivalently a clustering technique can be employed for graphs that do not comply with the $\Omega(1)$-model assumption. Clustering for the purpose of ad-hoc routing has been proposed by various researchers [19, 62]. A closely related approach is the construction of *dominating sets* [3, 6, 35, 40, 42, 46, 49, 52, 64, 77, 111], for instance for employment as routing backbones.

So far, the most popular network structure to model ad hoc networks has been the unit disk graph. The underlying assumption of this model is that the nodes are placed in the plane, all of them having the same transmission range—normalized to a radius of one unit of length. A more general model is provided by disk graphs where in contrast to unit disk graphs, nodes can have different transmission ranges. Disk graphs have also been widely used, but, while for unit disk graphs a number of theoretical results have been achieved, most of the knowledge on disk graphs is based on simulations. If disk graphs provide a simple method to analyze unidirectional links, it is not possible to model any kind of obstacles. In Chapter 10 we will go beyond unit disk graphs by allowing that certain sufficiently long edges may or may not exist in the considered network graph [67]. A model has been described in [7] which is—up to scaling—identical to our quasi unit disk graph model. [7] focused on geographic routing with guaranteed message delivery for certain instances of the quasi unit disk graph model. In Chapter 10 we will generalize and extend these results towards algorithm efficiency.

Flooding—an essential ingredient of many ad hoc routing algorithms—is one of the main techniques employed in Chapter 10. It is therefore crucial to reduce the number of messages sent. One way to reduce the cost of flooding is to lower the complexity of the network by using appropriate topology control mechanisms. Apart from this, there are other approaches which try to optimize flooding performance by using geographic information about the destination [8, 59]. These algorithms differ from the greedy routing/flooding approach presented in Chapter 10 in that they only try to flood into the right direction without actually applying geographic routing whenever possible.

Chapter 3

Models and Preliminaries

Models are to be used, not believed.
Henri Theil (1924–2000), in 'Principles of Econometrics'

At the beginning of every theoretical analysis stands the question of how to model the considered system. An obvious abstraction of a communication network is a graph with nodes representing networking devices and edges standing for network connections. The study of ad hoc networks in the geographic routing part of this thesis assumes that network nodes are placed in the Euclidean plane. Unless stated otherwise we furthermore model ad hoc networks as unit disk graphs [24]. A unit disk graph (UDG) is defined as follows:

Definition 3.1. (Unit Disk Graph) *Let $V \subset \mathbb{R}^2$ be a set of points in the 2-dimensional plane. The graph with edges between all nodes with distance at most 1 is called the unit disk graph of V.*

Accordingly, a unit disk graph models a flat environment with network devices equipped with wireless radio, all having equal transmission ranges. Edges in the UDG correspond to radio devices positioned in direct mutual communication range. Clearly, the unit disk graph model forms a highly idealistic abstraction of ad hoc networks. In Chapter 10 we will discuss routing in a model that more closely captures the connectivity characteristics of wireless networks.

To measure the quality of a routing algorithm, we attribute to each edge e a cost which is a function of the Euclidean length of e.

Definition 3.2. (Cost Function) *A cost function $c\colon]0,1] \mapsto \mathbb{R}^+$ is a non-decreasing function which maps any possible edge length d $(0 < d \leq 1)$ to a positive real value $c(d)$ such that $d' > d \Longrightarrow c(d') \geq c(d)$. For the cost of an edge $e \in E$ we also use the shorter form $c(e) := c(d(e))$.*

Note that $]0,1]$ really is the domain of a cost function $c(\cdot)$, that is, $c(\cdot)$ has to be defined for all values in this interval and in particular, $c(1) < \infty$. The cost

model thus defined includes all popular cost measures such as the link (or hop) distance metric ($c_\ell(d) :\equiv 1$), the Euclidean distance metric ($c_d(d) := d$), energy ($c_E(d) := d^2$, or more generally d^α for $\alpha \geq 2$), as well as hybrid measures which are positive linear combinations of the above metrics.

For convenience we also define the cost of a path, a sequence of contiguous edges, and of algorithms. The cost $c(p)$ of a path p is defined as the sum of the cost values of its edges. Analogously, the cost $c(\mathcal{A})$ of an algorithm $\mathcal{A}$ is defined as the summed up cost of all edges which are traversed during the execution of an algorithm on a particular graph. The question whether a node can send a message to several neighbors simultaneously does—unless noted otherwise—not affect our results, as the considered algorithms do not send messages in parallel to more than one recipient. An exception to this will be discussed in Chapter 10.

For the sake of simplicity we assume that the distance between any two nodes may not be arbitrarily small:

Definition 3.3. ($\Omega(1)$-model) *If the distance between any two nodes is bounded from below by a term of order $\Omega(1)$, i.e. there is a positive constant d_0 such that d_0 is a lower bound on the distance between any two nodes, this is referred to as the $\Omega(1)$-model.*

Graphs with this restriction have also been called *civilized* [30] or *λ-precision* [50] graphs in the literature. As a consequence of the $\Omega(1)$-model, the above-mentioned three metrics are equivalent up to a constant factor with respect to the cost of a path. As shown in the following lemma, this holds for all metrics defined according to Definition 3.2.

Lemma 3.1. *Let $c_1(\cdot)$ and $c_2(\cdot)$ be cost functions according to Definition 3.2 and let G be a unit disk graph in the $\Omega(1)$-model. Further let p be a path in G. We then have*

$$c_1(p) \leq \alpha \cdot c_2(p)$$

for a constant α.

Proof. Assume without loss of generality that p consists of k edges, that is, $c_\ell(p) = k$. As $d_0 \leq c_d(e) \leq 1$ for all edges $e \in E$ and the cost functions being nondecreasing, we have $c_1(p) \leq c_1(1) \cdot k$ and $c_2(d_0) \cdot k \leq c_2(p)$. Since—according to Definition 3.2—both $c_1(1)$ and $c_2(d_0)$ are constants greater than 0, the lemma holds with $\alpha = c_1(1)/c_2(d_0)$. □

Also the distance in a graph of a pair of nodes u and v—defined to be the cost of the shortest path connecting u and v—differs only by a constant factor for the different cost metrics:

Lemma 3.2. *Let G be a unit disk graph with node set V in the $\Omega(1)$-model. Further let $s \in V$ and $t \in V$ be two nodes and let p_1^* and p_2^* be optimal paths from s to t on G with respect to the metrics induced by the cost functions $c_1(\cdot)$ and $c_2(\cdot)$, respectively. It then holds that*

$$c_1(p_2^*) \leq \alpha \cdot c_1(p_1^*) \text{ and } c_1(p_2^*) \geq \beta \cdot c_1(p_1^*)$$

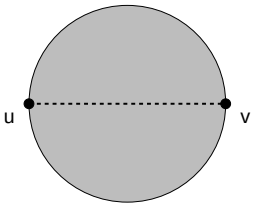

Figure 3.1: An edge (u, v) in the Gabriel Graph exists if and only if the shaded disk (including its boundary) does not contain any third node.

for two constants α and β, that is, the cost values of optimal paths for different metrics only differ by a constant factor.

Proof. By the optimality of p_2^* we have

$$c_2(p_2^*) \leq c_2(p_1^*). \tag{3.1}$$

Applying Lemma 3.1 we obtain

$$c_1(p_2^*) \leq \gamma \cdot c_2(p_2^*) \text{ and } c_2(p_1^*) \leq \delta \cdot c_1(p_1^*) \tag{3.2}$$

for two constants γ and δ. Combining Equations (3.1) and (3.2) yields $c_1(p_2^*) \leq \alpha \cdot c_1(p_1^*)$ for $\alpha = \gamma \cdot \delta$. Furthermore, by the optimality of p_1^*, we have $c_1(p_2^*) \geq c_1(p_1^*)$ and therefore the second equation of the lemma holds with $\beta = 1$. □

As this equivalence of cost metrics applies not only to the link, the Euclidean, and the energy metrics, but to all cost functions according to Definition 3.2, we sometimes refer to the "cost" of an edge and mean any cost metric belonging to the above class of cost functions. In Chapter 9 we show that employing clustering techniques a similar result can be achieved without the $\Omega(1)$-model assumption. Chapter 9 also describes the existence of two classes of cost functions and discusses their implications on routing. In the following chapters we will however adhere to the $\Omega(1)$-model for simplicity.

For our routing algorithms the network graph is required to be *planar*, that is without intersecting edges.[1] A planar graph features *faces*, contiguous regions separated by the edges of the graph. In order to achieve planarity on the unit disk graph G, we employ the *Gabriel Graph*. A Gabriel Graph contains an edge between two nodes u and v if and only if the disk (including its boundary) having $\overline{uv}$ as a diameter does not contain a "witness" node w (cf. Figure 3.1). Besides being planar, GG_G, the Gabriel Graph on the unit disk graph G, features two important properties:

[1] More precisely, the considered planar graphs are planar *embeddings* in the Euclidean plane.

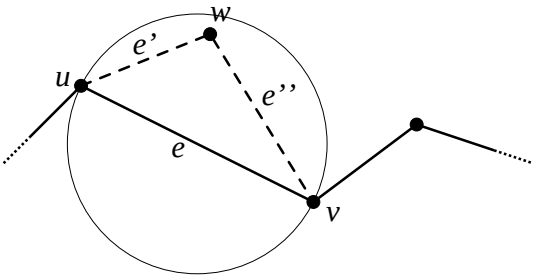

Figure 3.2: The Gabriel Graph contains an energy-optimal path.

- It can be computed locally: A network node can determine all its incident nodes in GG_G by mere inspection of its neighbors' locations (since G is a unit disk graph).

- The Gabriel Graph is a constant-stretch spanner for the energy metric: The construction of the Gabriel Graph on G preserves an energy-minimal path between any pair of network nodes. Together with the $\Omega(1)$-model it follows that the distance in GG_G between any pair of nodes is equal (up to constant factors) to their distance in G for all considered metrics. This is shown in the following lemma.

Lemma 3.3. *In the $\Omega(1)$-model the shortest path for any of the metrics according to Definition 3.2 on the Gabriel Graph intersected with the unit disk graph is only by a constant longer than the shortest path on the unit disk graph for the respective metric.*

Proof. We first show that at least one best path with respect to the energy metric on the UDG is also contained in $\mathrm{GG} \cap \mathrm{UDG}$. Suppose that $e = (u, v)$ is an edge of an energy optimal path p on the UDG. For the sake of contradiction suppose that e is not contained in $\mathrm{GG} \cap \mathrm{UDG}$. Then there is a node w in or on the circle with diameter $\overline{uv}$ (see Figure 3.2). The edges $e' = (u, w)$ and $e'' = (v, w)$ are also edges of the UDG and because w lies in the described circle, we have $e'^2 + e''^2 \leq e^2$. If w is inside the circle with diameter $\overline{uv}$, the energy for the path $p' := p \setminus \{e\} \cup \{e', e''\}$ is smaller than the energy for p and p is therefore not an energy-optimal path, contradicting the above assumption. If w lies exactly on the above circle, p' is an energy-optimal path as well and the argument applies recursively.

According to the optimality of $p^*_{\mathrm{GG}\cap\mathrm{UDG}}$, a shortest path on $\mathrm{GG} \cap \mathrm{UDG}$ with respect to a cost function $c(\cdot)$, we have $c(p^*_{\mathrm{GG}\cap\mathrm{UDG}}) \leq c(p) \leq \alpha \cdot c_E(p)$ for a constant α, the last inequality holding due to Lemma 3.1. Employing Lemma 3.2, we furthermore obtain $c_E(p) \leq \beta \cdot c(p^*_{UDG})$ for a constant β, where

p^*_{UDG} is a shortest path with respect to $c(\cdot)$ on the unit disk graph, which concludes the proof. □

Unless stated otherwise, we assume that every node locally computes its neighbors in the Gabriel Graph prior to the start of routing algorithms.

The geographic ad hoc routing algorithms we consider in this first part of the thesis can be defined as follows.

Definition 3.4. (Geographic Ad Hoc Routing Algorithm) *Let $G = (V, E)$ be a Euclidean graph. The task of a geographic ad hoc routing algorithm $\mathcal{A}$ is to transmit a message from a source $s \in V$ to a destination $t \in V$ by sending packets over the edges of G while complying with the following conditions:*

- *All nodes $v \in V$ know their geographic positions as well as the geographic positions of all their neighbors in G.*
- *The source s is informed about the position of the destination t.*
- *The control information which can be stored in a packet is limited by $\mathrm{O}(\log n)$ bits, that is, only information about a constant number of nodes is allowed.*
- *Except for the temporary storage of packets before forwarding, a node is not allowed to maintain any information.*

In the literature, geographic ad hoc routing has been given various other names, such as O(1)-memory routing algorithms in [13, 12], local routing algorithms in [61], geometric, position-based, or location-based routing. Due to these storage restrictions, geographic ad hoc routing algorithms are inherently local. In particular, nodes do not store any routing tables, eliminating a possible source of outdated information.

Finally, we assume that routing takes place much faster than node movement: A routing algorithm is modeled to run on temporarily stationary nodes. The issues faced when easing or giving up this assumption will be discussed in Chapter 12.

Chapter 4

Greedy Routing

Seek not happiness too greedily, and be not fearful of happiness.
Lao Tse

The probably most straightforward approach to geographic routing—which has also been studied as the first type of geographic routing algorithms in the related work—is *greedy forwarding*: Every node relays the message to be routed to its neighbor located "best" with respect to the destination. If "best" is interpreted as "closest to the destination", greedy forwarding can be formulated as follows:

Greedy Routing GR

0. Start at s.
1. Proceed to the neighbor closest to t.
2. Repeat step 1 until either reaching t or a local minimum with respect to the distance from t, that is a node v without any neighbor closer to t than v itself.

This formulation clearly reflects the simplicity of such an approach with respect to both concept and implementation. However, as indicated in Step 2 of the algorithm, it shows a big drawback: It is possible that the message runs into a "dead end", a node without any "better" neighbor. If backtracking techniques can overcome local minima in some cases, they fail to serve as a general solution to this problem, especially together with the strict message size limitations imposed on geographic routing (cf. Definition 3.4). Also alternative interpretations of "best neighbor" fail to reach the destination; in a "least deviation angle" approach for instance the message can end up in an infinite path loop [61].

If greedy routing however reaches the destination, it generally does so efficiently. Informally, this is due to the fact that—except in degenerate cases—the

message stays relatively close to the line connecting the source and the destination. As shown later in Chapter 8, employment of greedy routing whenever possible is beneficial above all in densely populated average-case networks. But also in worst-case networks the cost expended by greedy routing cannot become arbitrarily high:

Lemma 4.1. *If GR reaches t, it does so with cost $\mathrm{O}(d^2)$, where $d := |\overline{st}|$ denotes the Euclidean distance between s and t.*

Proof. This lemma has already been proved in [43]. For completeness we give an outline of a possible proof. Let $p := v_1, \ldots, v_k$ be the sequence of nodes visited during greedy routing. According to the definition of greedy routing, no two nodes v_i, v_j with odd indices i, j are neighbors. Further, since the distance to t is decreasing along the path p, all nodes v_i are inside $D(t, d)$, the disk with center t and radius d. $D(t, d)$ contains at most $\mathrm{O}(d^2)$ nodes with pairwise distance at least 1. It follows that p consists of $\mathrm{O}(d^2)$ nodes. □

In the following chapters, greedy routing will be employed as a routing algorithm component for its efficiency in both worst-case and average-case networks.

Chapter 5

Facing Dead Ends With Faces

I never forget a face, but in your case I'll be glad to make an exception.
Groucho Marx (1890–1977)

In the previous chapter we observed that greedy routing is not guaranteed to always reach the destination. This chapter introduces a type of geographic routing that, in contrast, always finds the destination if the network contains a connection from the source: routing based on faces.

5.1 Face Routing

The first geographic routing algorithm to be guaranteed to reach the destination was Face Routing introduced in [61]. Although we will formally describe a variant of Face Routing slightly adapted for our purposes, we will now give a brief overview of the original Face Routing algorithm.

At the heart of Face Routing lies the concept of faces, contiguous regions separated by the edges of a planar graph, that is a graph containing no two intersecting edges. The algorithm proceeds by exploration of face boundaries employing the local *right hand rule* in analogy to following the right hand wall in a maze (cf. Figure 5.1). On its way around a face, the algorithm keeps track of the points where it crosses the line $\overline{st}$ connecting the source s and the destination t. Having completely surrounded a face, the algorithm returns to the one of these intersections lying closest to the destination. From here, it proceeds by exploring the next face closer to t. If the source and the destination are connected, Face Routing always finds a path to the destination. It thereby takes at most $\mathrm{O}(n)$ steps, where n is the total number of nodes in the network.

5.2 AFR

Where the Face Routing algorithm can take up to $O(n)$ steps to reach the destination irrespective of the actual distance between the source and the destination in the given network, the main contribution of the Adaptive Face Routing algorithm AFR—as we presented it in [66]—consists in limiting the expended cost with respect to the length of the shortest path between s and t. Although the results discussed in the subsequent sections of this chapter go beyond AFR, we will first provide a summary of this algorithm for completeness and to give an overview of the employed technique.

As mentioned, the main problem with respect to the performance of Face Routing lies in the necessity of exploring the *complete* boundary of faces. It is thus impossible to bound the cost of this algorithm by the cost of an optimal path between s and t. If, however, we know the length of an optimal path connecting the source and the destination, Face Routing can be extended to *Bounded Face Routing BFR*: The exploration of faces is restricted to a searchable area, in particular an ellipse whose size is chosen such that it contains a complete optimal path. If the algorithm hits the ellipse, it has to "turn back" and continue its exploration of the current face in the opposite direction until hitting the ellipse for the second time, which completes the exploration of the current face. Briefly put—the details will be explained later—, since BFR does not traverse an edge more than a constant number of times, and since the bounding ellipse (together with the $\Omega(1)$-model and graph planarity) does not contain more than $O\big(|\overline{st}|^2\big)$ edges, the cost of BFR is in $O\big(c^2(p^*)\big)$, where p^* is an optimal path connecting s and t.

In most cases, however, a prediction of the length of an optimal path will not be possible. The solution to this problem finally leads to *Adaptive Face Routing AFR*: BFR is started with the ellipse size set to an initial estimate of the optimal path length. If BFR fails to reach the destination, which will be reported to the source, BFR will be restarted with a bounding ellipse of doubled size. (It is also possible to double the ellipse size directly without returning to the source.) If s and t are connected, AFR will eventually find a path to t. This iteration is asymptotically dominated by the cost of the algorithm steps performed in the last ellipse, whose area is at the most proportional to the squared cost of an optimal path. Consequently, also the cost of AFR is bounded by $O\big(c^2(p^*)\big)$.

Chapter 6 will show that in a lower bound graph no local geographic routing algorithm can perform better: AFR is asymptotically optimal.

5.3 OAFR

As described in Chapter 4, greedy routing promises to find the destination with low cost in all cases where it arrives at the destination. A natural approach to leveraging the potential of greedy routing above all for practical purposes therefore consists in combining greedy routing and face routing. In a first attempt

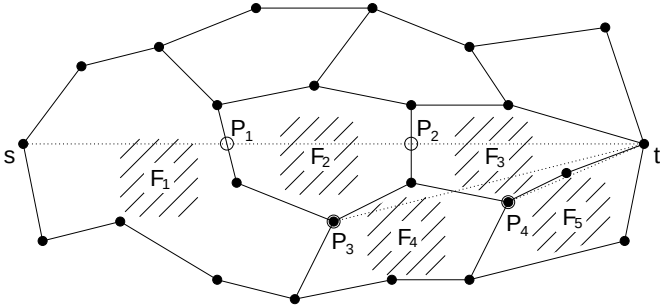

Figure 5.1: Face Routing starts at s, explores face F_1, finds P_1 on $\overline{st}$, explores F_2, finds P_2, and switches to F_3 before reaching t. OFR, in contrast, finds P_3, the point on F_1's boundary closest to t, continues to explore F_4, where it finds P_4, and finally reaches t via F_5.

we can literally combine Face Routing and AFR: Proceed in a greedy manner and use AFR to escape from potential local minima (an algorithm we will later call GAFR). We will however show in Chapter 8 that, employing greedy routing, this algorithm loses AFR's asymptotic optimality. Nevertheless we found a variant of AFR (OAFR) whose combination with greedy routing does finally yield algorithms (GOAFR and GOAFR$^+$) that are both average-case efficient and asymptotically optimal.

Similarly to the above description of AFR, we will explain our algorithm OAFR in three steps: OFR, OBFR, and OAFR.

Other Face Routing OFR differs from Face Routing in the following way: Instead of changing to the next face at the "best" *intersection* of the face boundary with $\overline{st}$, OFR returns—after completing the exploration of the boundary of the current face—to the boundary point (or one of the points) closest to the destination (Figure 5.1). Conserving the headway made towards the destination on each face, OFR in a sense uses a more natural approach than Face Routing.

Other Face Routing OFR

0. Begin at s and start to explore the face F containing the connecting line $\overline{st}$ in the immediate environment of s.

1. Explore the complete boundary of the face F based on local decisions employing the *right hand rule*.

2. Having accomplished F's exploration, advance to the point p closest to t on F's boundary. Switch to the face containing $\overline{pt}$ in p's environment and continue with step 1. Repeat these two steps until reaching t.

The number of steps taken by OFR is bounded as shown in the following lemma:

Lemma 5.1. *OFR always terminates in* $O(n)$ *steps, where* n *is the number of nodes. If* s *and* t *are connected, OFR reaches* t*; otherwise, disconnection will be detected.*

Proof. Let $F_1, F_2, \ldots, F_k$ be the sequence of the faces visited during the execution of OFR. We will first assume s and t to be connected. Since the switch between two faces always happens at the point on the face boundary closest to t and because the next face is chosen such that it always contains points which are nearer to t, no face is visited twice. Let further $p_0, p_1, p_2, ..., p_t$ be the *trace* of OFR's execution, where $p_i, i \geq 1$ is the point with minimum distance from t on the boundary of F_i. Because no face is visited more than once, we have that $\forall i > j : |\overline{p_i t}| < |\overline{p_j t}|$. Hence, if s and t are connected, we eventually arrive at a face with t on its boundary. (Otherwise, there is an i for which $p_i = p_{i+1}$, which means that the graph is disconnected.)

Since each face is explored at most once, each edge is visited at most four times. As every planar graph corresponds to the projection of a polyhedron on the plane, Euler's polyhedron formula can be employed: $n - m + f = 2$, where n, m, and f stand for the number of nodes, edges, and faces in the graph, respectively. Furthermore, the observations that (for $n > 3$) every face is delimited by at least three edges and that each edge is adjacent to at most two faces yield $3f \leq 2m$. Using Euler's formula we have $3m - 3n + 6 = 3f \leq 2m$ and therefore $m \leq 3n - 6$. Thus, OFR terminates after $O(n)$ steps. □

If the algorithm detects graph disconnection (finding $p_i = p_{i+1}$ for some $i \geq 0$), this can be reported to the source by again using OFR in the reverse direction.

Remark (Gabriel Graph) When applying OFR on a Gabriel Graph—as we will do for the routing on unit disk graphs—OFR can be simplified in the following way: Instead of changing faces at the *point* on the face boundary which is closest to t, it is possible to take the *node* which is closest to t. This modification leaves the property described in Lemma 5.1 unchanged, as also the modified OFR algorithm always switches to a new face in Step 2 if it is run on the Gabriel Graph. This is illustrated in Figure 5.2. As definitions and explanations become clearer, we will use this modified form of the OFR algorithm for the description of the subsequent algorithms. Equivalent results can be achieved with the original version of the algorithm.

When trying to formulate a statement on OFR's cost, the main problem arising is its traversal of complete boundaries of faces: Informally put, OFR can meet an incredibly big face whose total exploration is prohibitively expensive compared to an optimal path from s to t. In order to solve this, we borrow

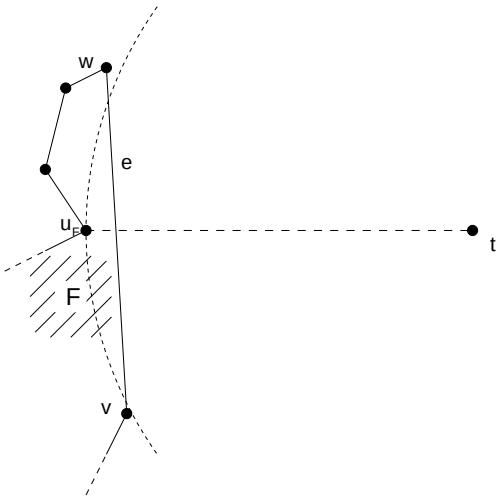

Figure 5.2: Also OFR modified to switch to the next face at the *node* closest to t (instead of the *point* closest to t) progresses with each face switch if it runs on the Gabriel Graph. In particular it cannot happen that the algorithm is caught in an infinite loop: Having arrived at u_F, the node of face F located closest to t, the algorithm always switches to a face other than F. The only possible way of constructing a counterexample fails: A constellation forcing the modified OFR algorithm to again select F as the next face at u_F has at least one edge $e = (v, w)$ on F's boundary intersecting the line segment $\overline{u_F t}$; otherwise t would lie inside F, implying that s and t would be disconnected. The fact that both v and w are not closer to t than u_F—u_F is the node on F's boundary closest to t—implies that at least one of u_F and t are located within the disk with diameter $\overline{vw}$, which contradicts the existence of the edge e in the Gabriel Graph.

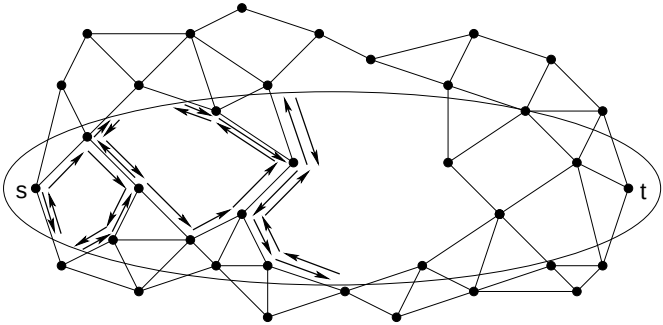

Figure 5.3: Execution of OBFR if the ellipse is not chosen sufficiently large.

AFR's trick to bound the searchable area by an ellipse containing an optimal path. Consequently we obtain *Other Bounded Face Routing OBFR.*

For the sake of simplicity we assume for the following description of OBFR that s and t are connected. If $\tilde{c}$ is an estimate of the Euclidean length of a shortest path between s and t, let $\mathcal{E}$ be the ellipse with foci s and t and with the length of the major axis being $\tilde{c}$ (in other words, $\mathcal{E}$ contains all paths from s to t of Euclidean length at most $\tilde{c}$).

Other Bounded Face Routing OBFR

0. Step 0 of OFR.

1. Step 1 of OFR, but do not leave $\mathcal{E}$: When hitting $\mathcal{E}$, continue the exploration of the current face F in the opposite direction. F's exploration will afterwards be complete when hitting $\mathcal{E}$ for the second time.

2. Step 2 of OFR with one modification: If the node closest to t on F's boundary is the same one as in the previous iteration, that is, no progress has been made in Step 1, report failure back to s by means of OBFR.

Figures 5.3 and 5.4 illustrate the execution of OBFR if the ellipse is chosen too small and if the ellipse contains a path from s to t, respectively. The cost expended by OBFR can be bounded as follows:

Lemma 5.2. *If the length $\tilde{c}$ of the major axis of $\mathcal{E}$ is at least the length of a—with respect to the Euclidean metric—shortest path between s and t, OBFR reaches the destination. Otherwise OBFR reports failure to the source. In any case, OBFR expends cost at most $\mathrm{O}(\tilde{c}^2)$.*

Proof. We first assume that $\tilde{c}$ is at least the length of a shortest (Euclidean) path p^*, that is, p^* is completely contained in $\mathcal{E}$. Since OBFR stays within $\mathcal{E}$

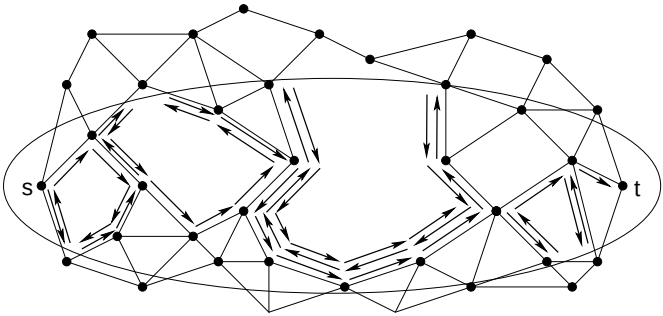

Figure 5.4: Execution of OBFR if the ellipse is chosen large enough to contain a path from s to t.

while routing a message, we only look at the part of the graph which lies inside $\mathcal{E}$. We define the faces to be those contiguous regions which are separated by the edges of the graph and by the boundary of $\mathcal{E}$. (Hence, if a face is cut into several pieces by the boundary of $\mathcal{E}$, now each such piece is denoted a face.) Assume for the sake of contradiction that OBFR reports failure, that is, the algorithm does not make progress in Step 2. This is only possible if the currently traversed face boundary cuts the area enclosed by the ellipse into a region containing s and a second region containing t (cf. Figure 5.3). In this case however, $\mathcal{E}$ does not contain any path connecting s and t, which contradicts our assumption and therefore proves the first sentence of the lemma.

If no path connects s and t within $\mathcal{E}$, a face boundary separating s from t as described in the previous paragraph exists. This is detected by OBFR, making no progress beyond a node v. As OBFR reached v starting from s, $\mathcal{E}$ contains a path from v to s and OBFR can be restarted in the opposite direction with the same ellipse, eventually reaching s and reporting failure.

Finally, it remains to be shown that the cost expended does not exceed $\mathrm{O}(c^2)$. If $\mathcal{E}$ contains a path connecting s and t, every face is—for the same reasons as for OFR—visited at most once. Otherwise, every face is visited at most twice (the face where failure is detected can be visited an additional time). Furthermore, during the traversal of a face boundary, each edge can be visited at most four times. Consequently, any edge is traversed at most a constant number of times during the complete execution of OBFR.

Due to the planarity of the considered graph, the number of edges is linear in the number of nodes (cf. proof of Lemma 5.1). Furthermore—according to the employed $\Omega(1)$-model—the circles of radius $d_0/2$ around all nodes do not intersect each other. Since the length of the semimajor axis a of the ellipse $\mathcal{E}$ is

$\tilde{c}/2$, and since the area of $\mathcal{E}$ is smaller than πa^2, the number of nodes n' inside $\mathcal{E}$ is bounded by

$$n' \leq \frac{\pi a^2}{\pi \left(\frac{d_0}{2}\right)^2} = \frac{\tilde{c}^2}{d_0^2} \in \mathrm{O}\big(\tilde{c}^2\big)\,.$$

Having thus found an upper bound for the number of messages sent, the last statement of the lemma follows with Lemma 3.2 for all cost metrics defined according to Definition 3.2. □

Note that the above specification of OBFR omits an important point for clarity of representation: The algorithm can distinguish between the case where the chosen ellipse is not sufficiently large to contain a path connecting s with t and the case where s and t are disconnected. As described above, the first case is detected if no progress is made after hitting the ellipse. In the second case there also exists a node beyond which no progress is made; however, this node is detected as such without hitting the ellipse, that is, after traversing the complete boundary of the network component containing the source.

Since there is usually no a priori information on the optimal path length, we—in analogy to AFR—initially use a small estimate for the ellipse size and iteratively enlarge it until reaching the destination.

Other Adaptive Face Routing OAFR

0. Initialize $\mathcal{E}$ to be the ellipse with foci s and t the length of whose major axis is $2 \cdot |\overline{st}|$.

1. Start OBFR with $\mathcal{E}$.

2. If the destination has not been reached, double the length of $\mathcal{E}$'s major axis and go to step 1.

Exploiting that OBFR is able to distinguish between insufficient ellipse size and graph disconnection between s and t, also OAFR detects graph disconnection. Furthermore OAFR's cost is bounded as follows:

Theorem 5.3. *If s and t are connected, OAFR reaches the destination with cost $\mathrm{O}\big(c^2(p^*)\big)$, where p^* is an optimal path. If s and t are disconnected, OAFR detects so and reports to s.*

Proof. We denote the first estimate $\tilde{c}$ on the optimal path length by $\widetilde{c_0}$ and the consecutive estimates by $\widetilde{c_i} := 2^i \widetilde{c_0}$. Furthermore, we define k such that $\widetilde{c_{k-1}} < c_d(p^*) \leq \widetilde{c_k}$. For $c(\mathrm{OBFR}[\tilde{c}])$, the cost of OBFR with the length of $\mathcal{E}$'s major axis set to $\tilde{c}$, we have $c(\mathrm{OBFR}[\tilde{c}]) \in \mathrm{O}\big(\tilde{c}^2\big)$ and therefore

$$c_\ell(\mathrm{OBFR}[\tilde{c}]) \leq \lambda \cdot \tilde{c}^2$$

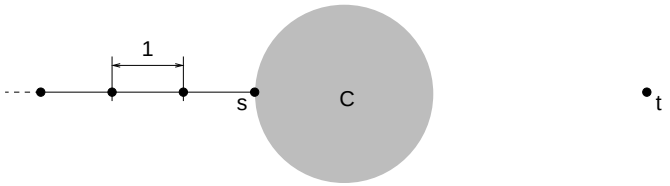

Figure 5.5: If $n'/2$ nodes are located in the node cluster C (represented as a gray disk) and $n'/2$ nodes form the spike on the left, OAFR executes its component OBFR $\Theta(\log n')$ times, each OBFR execution having cost in $\Theta(n')$ if the nodes in C are placed in a maze-like structure, before detecting that s and t are disconnected.

for a constant λ (and sufficiently large $\tilde{c}$). The total cost of OAFR can therefore be bounded by

$$\begin{aligned} c_\ell(\text{OAFR}) &\leq \sum_{i=0}^{k} c_\ell(\text{OBFR}[\tilde{c}_i]) \leq \sum_{i=0}^{k} \lambda \left(2^i \tilde{c}_0\right)^2 \\ &= \lambda \tilde{c}_0{}^2 \frac{4^{k+1}-1}{3} < \frac{16}{3} \lambda \left(2^{k-1} \tilde{c}_0\right)^2 \\ &< \frac{16}{3} \lambda \cdot c_{\mathrm{d}}^2(p^*) \in \mathrm{O}\big(c_{\mathrm{d}}^2(p^*)\big)\,. \end{aligned}$$

□

Remark It can be shown that for OAFR (and also AFR) the cost of detecting disconnectivity of s and t is bounded by $\mathrm{O}(n' \log n')$, where n' is the number of nodes in the network component containing s: The number of OBFR executions is at most in $\mathrm{O}(\log n')$, while the cost expended by OBFR in each of these executions is at most linear in n'. As illustrated in Figure 5.5, there are graphs for which OAFR expends cost $\Theta(n' \log n')$.

Having shown in this chapter that the cost expended by the OAFR algorithm is asymptotically upper-bounded by the square of the shortest path between the source and the destination, we will discuss the significance and the quality of this result in the following chapter.

Chapter 6

A Lower Bound

Egotist, n. A person of low taste, more interested in himself than in me.
Ambrose Bierce (1842–1914)

As presented in the previous chapter, the OAFR algorithm reaches the destination with cost $O(c^2)$, where c is the cost of the shortest path between the source and the destination. A natural question arising is whether this guarantee is good or if there are algorithms that can perform better. In the following we will give an answer to this question by showing that no geographic routing algorithm according to Definition 3.4 can find the destination with lower cost. In particular we give a constructive lower bound:

Theorem 6.1. *Let the cost of a best route for a given source-destination pair be c. There exist graphs where any deterministic (randomized) geographic ad hoc routing algorithm has (expected) cost $\Omega(c^2)$ for any cost metric according to Definition 3.2.*

Proof. We construct a family of networks as follows. We are given a positive integer k and define a Euclidean graph G (see Figure 6.1): On a circle we evenly distribute $2k$ nodes such that the distance between two neighboring points is exactly 1; thus, the circle has radius $r \approx k/\pi$. For every second node of the circle we construct a chain of $\lceil r/2 \rceil - 1$ nodes. The nodes of such a chain are arranged on a line pointing towards the center of the circle; the distance between two neighboring nodes of a chain is exactly 1. Node w is one arbitrary circle node with a chain: The chain of w consists of $\lceil r \rceil$ nodes with distance 1. The last node of the chain of w is the center node; the edge to the center node does not need to have length 1.

The unit disk graph consists of the edges on the circle and the edges on the chains only. In particular, there is no edge between two chains because all chains except the w chain end strictly outside radius $r/2$. The graph has k chains with $\Theta(k)$ nodes each.

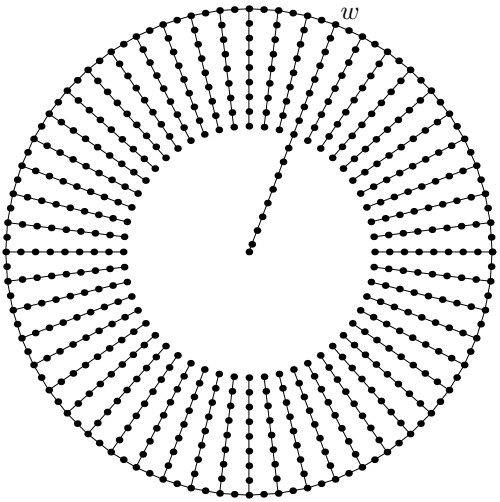

Figure 6.1: Lower bound graph.

We route from an arbitrary node on the circle (the source s) to the center of the circle (the destination t). An optimal route between s and t follows the shortest path on the circle until it hits node w, and then directly follows w's chain to t with link cost $c \leq k + r + 1 \in \mathrm{O}(k)$. A routing algorithm with routing tables at each node will find this best route.

A geographic routing algorithm, in contrast, needs to find the "correct" chain w. Since there is no routing information stored at the nodes, this can only be done by exploring the chains. Any deterministic algorithm needs to explore the chains in a deterministic order until it finds the chain w. Thus, an adversary can always place w such that w's chain will be explored as the last one. The algorithm will therefore explore $\Theta(k^2)$ (instead of only $\mathrm{O}(k)$) nodes.

The argument is similar for randomized algorithms. By placing w accordingly (randomly!), an adversary forces the randomized algorithm to explore $\Omega(k)$ chains before chain w with constant probability. Then the expected link cost of the algorithm is $\Omega(k^2)$.

As all edges (but one) in our construction have length 1, the cost values in the Euclidean distance, the link distance, and the energy metrics are equal. As for any fixed cost metric $c'(\cdot)$ according to Definition 3.2 $c'(1)$ is also a constant, the $\Omega(c^2)$ lower bound holds for all according metrics. □

Note that our lower bound holds generally, not only for $\Omega(1)$-graphs. As we will show in Chapter 9, however, a similar lower bound proves that in general graphs the cost metrics according to Definition 3.2 fall into two classes.

Given this lower bound, we can now state that OAFR is asymptotically optimal for unit disk graphs in the $\Omega(1)$-model.

Theorem 6.2. *Let c be the cost of an optimal path for a given source-destination pair on a unit disk graph in the $\Omega(1)$-model. In the worst case, the cost for applying OAFR to find a route from the source to the destination is $\Theta(c^2)$. This is asymptotically optimal.*

Proof. This theorem is an immediate consequence of Theorems 5.3 and 6.1. □

Chapter 7

Combining Greedy and Face Routing

Skiing combines outdoor fun with knocking down trees with your face.
*Dave Barry (*1947)*

A greedy routing approach as presented in Chapter 4 is not only worth being considered due to its simplicity in both concept and implementation. Above all in dense networks such an algorithm can also be expected to find paths of good quality efficiently; here, the straightforwardness of a greedy strategy contrasts highly the inflexible exploration of faces inherent to face routing. For practical purposes it is inevitable to improve the performance of a face routing variant, for instance by leveraging the potential of a greedy approach.[1]

In this chapter we will present two algorithms combining greedy and face routing. The GOAFR algorithm introduces the principal concept employed to integrate these two approaches. Later, the GOAFR$^+$ algorithm is not only proved to be worst-case optimal but—due to its additional improvements—will be shown in Chapter 8 to outperform all previously proposed geographic routing algorithms in average-case networks.

7.1 GOAFR: Greedy OAFR

A possible combination of greedy routing and our OAFR algorithm forms *Greedy Other Adaptive Face Routing GOAFR* (pronounced as "gopher"). In principle

[1] Interestingly, simple non-geographic flooding of the network with exponentially increasing time-to-live values achieves the same worst-case efficiency as the face routing algorithms described in the previous chapter if we allow the network nodes to temporarily store flooding state information in some additional bits. Only additional average-case assessment as carried out later in Chapter 8 will display the main advantages of routing algorithms combining greedy and face routing over such simplistic approaches.

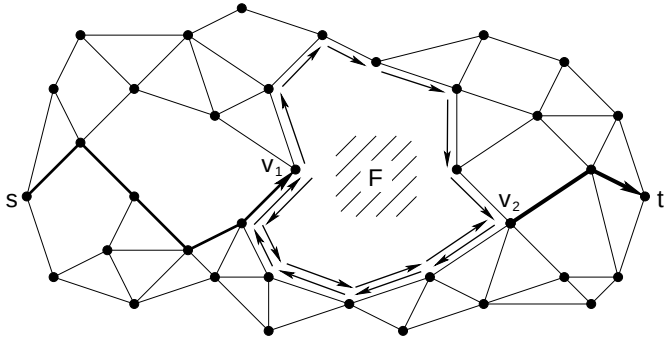

Figure 7.1: Starting at s, GOAFR proceeds in greedy mode until reaching the local minimum v_1. The algorithm switches to face routing mode and explores the boundary of face F to find v_2, the node closest to t on F's boundary. GOAFR falls back to greedy mode and finally reaches t. Note that GOAFR's ellipse has been omitted for simplicity.

greedy routing is used as long as possible. Local minima potentially met underway are escaped from by use of OAFR (Figure 7.1). After specifying the algorithm, we will show that GOAFR retains OAFR's asymptotic optimality.

The greedy steps taken in the following specification of the GOAFR algorithm correspond to the steps described as the Greedy Routing algorithm in Chapter 4.

Greedy Other Adaptive Face Routing GOAFR

0. Initialize $\mathcal{E}$ to be the ellipse with foci s and t the length of whose major axis is $2 \cdot |\overline{st}|$; start at s.

1. Perform greedy steps until either reaching t or a local minimum u_m. If the next step leads beyond $\mathcal{E}$, double the length of $\mathcal{E}$'s major axis. If reaching a local minimum, proceed with step 2 starting at u_m.

2. Execute OAFR on the first face F only. Double the length of $\mathcal{E}$'s major axis as long as necessary.

3. Terminate if OAFR reaches t. If OAFR detects disconnection, report so to s (by use of GOAFR). Otherwise (OAFR has found—after complete exploration of F's boundary within $\mathcal{E}$ in its current size—a node v with $|\overline{vt}| < |\overline{u_m t}|$) continue with step 1 at the node closest to t found by OAFR.

In order to show that the cost expended by GOAFR is bounded, we first prove that the maximal size of the ellipse $\mathcal{E}$ is related to the length of a shortest path between s and t.

Lemma 7.1. *If s and t are connected, the major axis of $\mathcal{E}$ in GOAFR will not exceed $2 \cdot \max(c_d(p^*), 3|\overline{st}|)$, where $c_d(p^*)$ is the Euclidean length of an optimal path (with respect to the Euclidean metric).*

Proof. According to the definition of an ellipse as the locus of all points having the same sum of distances from the two foci, the ellipse $\mathcal{E}$ with major axis length c contains all paths of Euclidean length at most c. With the doubling strategy in OAFR, the biggest ellipse employed in this "subalgorithm" of GOAFR will have a major axis not longer than $2 \cdot c_d(p^*)$. Additionally the Greedy Routing part of GOAFR might walk (almost) along the boundary (but is always on the inside) of $D(t, |\overline{st}|)$, the disk centered at t and with radius $|\overline{st}|$. $\mathcal{E}$ with major axis length $3|\overline{st}|$ is the smallest ellipse with s and t as foci which completely contains $D(t, |\overline{st}|)$. Combining the OAFR part, the GR part, and the fact that the ellipse might get twice as big as the smallest possible ellipse, the lemma follows. Note that these considerations hold with slight modifications for any (sufficiently small) start value and increment factor for the major axis length of the ellipse. If these two parameters are chosen as in the definition of GOAFR, the bound on the length of the major axis can be improved to $\max(2|p^*|, 4|\overline{st}|)$. □

In the following lemma we consider the execution of GOAFR within an ellipse of fixed size.

Lemma 7.2. *For an ellipse $\mathcal{E}$ with fixed major axis length c, GOAFR traverses at most $O(c^2)$ edges.*

Proof. GOAFR consists of face routing steps (more specifically OBFR) and of greedy routing steps. Each of these steps (a greedy step or the complete exploration of a face) brings us to a node closer to the destination t. The number of greedy steps is bounded by $O(c^2)$ (cf. Lemma 4.1). As in OBFR, each face is explored at most once and hence each (Gabriel Graph) edge is used at most four times in a face routing step. Combined, this proves the lemma. □

This leads to the conclusion that the cost of GOAFR is bounded by the squared cost of the optimal path:

Theorem 7.3. *If s and t are connected, GOAFR reaches t with cost $O(c^2(p^*))$. This is asymptotically optimal. If s and t are not connected, GOAFR reports so.*

Proof. If s and t are connected, the lemma is a consequence of Lemma 7.1 (using that $|\overline{st}| \leq c_d(p^*)$ and Lemma 7.2 combined with the fact that the ellipse axis lengths form a geometric sequence (cf. proof of Theorem 5.3) and the asymptotic equivalence of all cost metrics according to Definition 3.2. The lower bound discussed in Chapter 6 proves the asymptotic optimality. If s and t are not connected, the node detecting this (after a face routing step which does not yield a node closer to t) sends the result back to s by means of the same algorithm. □

Remark Similar to OAFR, GOAFR detects disconnectivity of s and t with cost at most $O(n' \log n')$, where n' is the number of nodes in the network component containing s.

7.2 GOAFR$^+$: Improving GOAFR

Having shown the concept of combining greedy and face routing, we will present in this section the GOAFR$^+$ algorithm (pronounced as "gopher-plus"), which forms a refinement of the GOAFR algorithm. The primary reason for the modifications resulting in GOAFR$^+$ is their leading to better performance in average-case networks, which will be shown in Chapter 8. At the same time, however, the asymptotic optimality of the GOAFR algorithm is preserved, as proved after the following specification of GOAFR$^+$.

7.2.1 The GOAFR$^+$ Algorithm

Similar to GOAFR, the GOAFR$^+$ algorithm is a combination of greedy routing and face routing. Whenever possible, the algorithm tries to route in a greedy manner; in order to overcome local minima with respect to the distance from the destination, face routing is employed.

In face routing mode, GOAFR$^+$ restricts the searchable area in a similar way as GOAFR. Our simulations (as discussed later in Chapter 8) showed that choosing a circle centered at the destination t instead of an ellipse and above all gradually reducing its radius while the message approaches t improves the average-case performance.

More importantly, for average-case considerations, the algorithm should fall back to greedy routing as soon as possible after escaping the local minimum. This is suggested by the observation that greedy forwarding is—especially in dense networks—more efficient than face routing in the average case. If GOAFR traverses the complete boundary of a face—once it is in face routing mode—, GOAFR$^+$ tries to avoid doing so and to return to greedy routing as early as possible. As will be illustrated in Chapter 8, this must not be done too simplistically—such as whenever the algorithm in face routing mode is closer to the destination than the escaped from local minimum—, as this would happen at the expense of the algorithm's asymptotic optimality. In order to preserve this property, the GOAFR$^+$ algorithm employs two counters p and q to keep track of how many of the nodes visited during the current face routing phase are located closer to the destination (counted with p) and how many are at least as far from the destination (counted with q) than the starting point of the current face routing phase; as soon as a certain fallback condition holds, GOAFR$^+$ continues in greedy mode.

Figure 7.2 provides an illustration to the following specification of the GOAFR$^+$ algorithm.

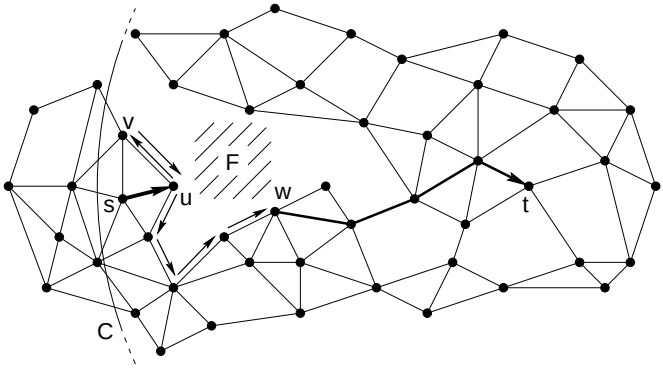

Figure 7.2: The GOAFR$^+$ algorithm starts from s in greedy mode. At node u, it reaches a local minimum, a node without any neighbors closer to t. GOAFR$^+$ switches to face routing mode and begins to explore the boundary of face F (in clockwise direction). At node v the algorithm hits the bounding circle C and turns back to continue the exploration of F's boundary in the opposite direction. After each step the counters p and q are updated. At node w the fallback condition $p > \sigma\, q$ holds ($p = 2, q = 4$ with the assumption $1/4 \leq \sigma < 1/2$); GOAFR$^+$ falls back to greedy mode and continues to finally reach t. (Gradual reduction of C's size during GOAFR$^+$'s execution is not shown.)

GOAFR$^+$ The algorithm parameters ρ_0, ρ, and σ are chosen prior to the start of the algorithm and remain constant throughout the execution. For the algorithm to work correctly, these parameters have to comply with the conditions $1 \leq \rho_0 < \rho$ and $0 < \sigma$.[2]

0. Begin at s. Initialize C to be the circle centered at t with radius $r_C := \rho_0\, |\overline{st}|$.

1. **(Greedy Routing Mode)** Repeat taking greedy steps until either reaching t or a local minimum. In the former case the algorithm terminates, in the latter case continue with step 2. Whenever possible, reduce C's radius ($r_C := r_C/\rho$) as long as the currently visited node stays within C.

2. **(Face Routing Mode)** Let u_i be the currently visited local minimum. Start exploring the boundary of F_i, the face containing the connecting line $\overline{u_i t}$ in the immediate environment of u_i. When completing F_i's exploration and returning to u_i, advance to the node visited so far closest to t and continue with step 1. If—after F's complete exploration—no visited node

[2]In our simulations $\rho_0 = 1.4$, $\rho = \sqrt{2}$, and $\sigma = \frac{1}{100}$ proved to be good choices for practical purposes.

is closer to t than u_i, report graph disconnection to s (using GOAFR$^+$). During the exploration of F_i's boundary use two counters p and q to keep track of the number of nodes visited on F_i's boundary: p counts the nodes closer to t than u_i and q the nodes not located closer to t than u_i. Take a special action if one of the following conditions holds:

2a. Hitting C for the first time, turn back and continue exploring F_i's boundary in the opposite direction.

2b. C is hit for the second time: If none of the visited nodes is closer to t than u_i, enlarge C ($r_C := \rho\, r_C$) and continue with step 2 as if having started from u_i. Otherwise advance to the node visited so far closest to t and continue with step 1.

2c. If $p > \sigma\, q$, that is, we have visited (up to a constant factor σ) more nodes on F_i's boundary closer to t than nodes not closer to t, advance to the node seen so far closest to t (if this is not the currently visited node) and continue with step 1.

With this description of the GOAFR$^+$ algorithm, we can now proceed to showing that its worst-case performance is asymptotically equivalent to the one of GOAFR.

7.2.2 GOAFR$^+$ is Asymptotically Optimal

GOAFR$^+$ uses a circle C centered at t to restrict itself to a searchable area. During the algorithm execution the radius r_C is adapted in predefined steps according to the current distance from t. In particular, the values potentially assumed by r_C form a geometric sequence $r_{C_i} = r_{max}\,(\frac{1}{\rho})^i, i = 0, ..., k$, where r_{max} depends on the length and the shape of the optimal path from s to t (cf. proof of Theorem 7.7) and ρ is one of GOAFR$^+$'s predefined constant algorithm parameters. Since r_C can both increase and decrease during algorithm execution, the steps taken in a circle C_i with radius r_{C_i} need not occur consecutively. In the following we consider the steps taken by the algorithm in a fixed circle C_i.

Lemma 7.4. *If s and t are connected within the circle C_i, GOAFR$^+$ reaches t. If s and t are not connected within C_i, GOAFR$^+$ detects so.*

Proof. We first assume there is a connection from s to t within C_i. For the definition of a *round* we distinguish three cases: According to the current algorithm execution, a *round* can be either a) a greedy step, b) a face routing phase terminated by early fallback, or c) a face routing phase terminated after exploration of the complete boundary of the current face and advancing to the node closest to t. We show that after every round the algorithm is closer to t than before that round: This holds in case a) since a greedy step can only reduce the distance to t, and in case b), as the fallback condition can only hold immediately after incrementing the counter p (that is after visiting at least one

node closer to t) and since the algorithm then advances to the node seen so far closest to t; in case c), the algorithm approaches t since the boundary of the currently explored face—this face contains in its interior points closer to t than where this round started—contains a point closer to t if and only if there is a connection to t. (Note that again the remark made after Lemma 5.1 holds: Graphs can be constructed where a face F's boundary contains *points* but not *nodes* that are closer to t than a given boundary node, in which case the algorithm could fail. Since we employ the Gabriel Graph, such cases can however not occur: The algorithm can forward to a face boundary's *node* closest to t.) As the algorithm reduces the distance to the destination in each round, it finally reaches t.

If s and t are not connected within C_i, GOAFR$^+$—in face routing mode—either hits C_i twice without finding a node closer to t (in which case the algorithm will continue on a bigger circle, which is beyond the scope of this lemma), or it explores the complete boundary of the current face (cf. above case c)) without finding a node closer to t, which is the case if and only if s and t are not connected at all. □

Having shown that GOAFR$^+$ reaches the destination, we can now prove in several steps that its cost is bounded. We first consider the cost expended when traversing the boundary of a given face within a circle of fixed size.

Lemma 7.5. *Let $c'_F(GOAFR^+)$ be the cost of all face routing steps taken when exploring the boundary of face F within the circle C_i. $c'_F(GOAFR^+)$ is less than $\gamma\, c_F$ for a constant γ and with c_F being the total cost of traversing F's boundary once.*

Proof. We first show that the lemma holds for the link distance metric, $c(e) \equiv 1$ for any edge e: The total number of edges traversed by GOAFR$^+$ when exploring F is less than γc_{ℓ_F}, where c_{ℓ_F} is the number of edges traversed when traveling around F once.

We assume that the boundary of face F is involved in k face routing rounds, and that for $1 \leq j \leq k$, s_j is the node where round j is started. Let p_j and q_j be the final values of the counters p and q, respectively, in round j. According to the fallback condition in step 2c of the algorithm we have $p_j > \sigma\, q_j$. Let P_j and Q_j be the sets of nodes visited in round j closer to t than s_j and at least as far from t as s_j, respectively. Since a node can be counted for a second time after hitting C_i, we have $|P_j| \leq p_j \leq 2\,|P_j|$ and $|Q_j| \leq q_j \leq 2\,|Q_j|$. Furthermore we define N_j to be the set of nodes newly visited in round j. Since after each round—a greedy step or the exploration of a face—the algorithm is strictly closer to t than before that round, all nodes closer to t must be newly visited ones, that is $P_j \subseteq N_j$. Since we also have to account for the steps taken by the algorithm possibly proceeding—once the fallback criterion holds—to the node seen so far closest to t, the number of steps taken in round j is not greater than

$2\,(p_j + q_j)$. In summary we obtain for the total cost of the algorithm exploring F's boundary:

$$\begin{aligned}\sum_{j=1}^{k} 2\,(p_j + q_j) \;&<\; \sum_{j=1}^{k} 2\,(1+\frac{1}{\sigma})\,p_j \;\leq\; \sum_{j=1}^{k} 4\,(1+\frac{1}{\sigma})\,|P_j| \\ &\leq\; \sum_{j=1}^{k} 4\,(1+\frac{1}{\sigma})\,|N_j| \;\leq\; 4\,(1+\frac{1}{\sigma})\,c_{\ell_F},\end{aligned}$$

the last step following from $\sum_{j=1}^{k} |N_j| \leq c_{\ell_F}$.

If the fallback criterion never holds during F's exploration (which is only possible in the final round for F), the algorithm traverses F's complete boundary and advances to the node closest to t, which incurs additional cost less than $2\,c_{\ell_F}$.

The lemma holds for the link distance metric. Since the algorithm is assumed to run in the $\Omega(1)$-model, the lemma also holds for any other cost metric (cf. Chapter 3). □

With this lemma, the overall cost—in greedy and in face routing mode—expended in a given circle of fixed size can be bounded:

Lemma 7.6. *The total cost of the steps taken by GOAFR$^+$ within the circle C_i with radius r_{C_i} is in $\mathrm{O}\big(r_{C_i}^2\big)$.*

Proof. According to the previous lemma we have $c'_F(\mathrm{GOAFR}^+) \leq \gamma\, c_F$ for all steps performed in face routing mode. Summing up over all faces in C_i, we obtain

$$\sum_{F \in C_i} c'_F(\mathrm{GOAFR}^+) \;<\; \gamma \cdot \sum_{F \in C_i} c_F \;\leq\; \gamma \cdot 2 \sum_{e \in C_i} c(e),$$

the last step following from the fact that each edge e is adjacent to at most two faces. To account for the greedy steps, we add another $\sum_{e \in C_i} c(e)$ in order to obtain the total cost expended in C_i since any edge can be traversed at most once in greedy mode (each round—a greedy step or the exploration of a face—taking the algorithm strictly closer to t). Since we employ a planar graph, with the fact that (in a graph with more than three edges) each face is adjacent to at least three edges and using the Euler polyhedral formula, we obtain that $|E_i| \in \mathrm{O}(|V_i|)$, where $|E_i|$ is the number of edges and $|V_i|$ the number of nodes in C_i (cf. proof of Lemma 5.1). The lemma finally follows with $\sum_{e \in C_i} c(e) \in \mathrm{O}(|E_i|)$—resulting from the equivalence of the link distance metric with any other metric in the $\Omega(1)$-model (cf. Chapter 3)—and the fact that in the $\Omega(1)$-model the number of nodes in a given region is proportional to its size (cf. proof of Lemma 5.2). □

As described above, GOAFR$^+$ employs a set of bounding circles whose radii form a geometric sequence. This together with the fact that the maximum radius is bounded by the Euclidean length of an optimal path from s to t, leads to the following theorem.

Theorem 7.7. *Let p^* be an optimal path from s to t. In the $\Omega(1)$-model, GOAFR$^+$ reaches t with cost $\mathrm{O}\big(c^2(p^*)\big)$ if s and t are connected, which is asymptotically optimal. If s and t are not connected, GOAFR$^+$ reports so to the source.*

Proof. Let $c_d(p^*)$ be the Euclidean length of a shortest path from s to t. If s and t are connected, the circle centered at t and with radius $c_d(p^*)$ completely contains p^*. Since GOAFR$^+$ only enlarges the bounding circle if it does not contain a path from s to t, and according to GOAFR$^+$'s radius update policy with the constant factor ρ, the maximum radius reached is smaller than $\rho\, c_d(p^*)$. In order to compute the total cost of the algorithm, we add up the cost expended in each used circle. According to Lemma 7.6 it is sufficient to consider the sum of the squared radii of all employed circles. Let r_{max} be the radius of the largest used circle. For some $k \geq 0$ the areas of all used circles sum up to

$$\begin{aligned}
\sum_{i=0}^{k} \left[r_{max} \cdot \left(\frac{1}{\rho}\right)^i \right]^2 &= \frac{1 - \left(\frac{1}{\rho}\right)^{2(k+1)}}{1 - \left(\frac{1}{\rho}\right)^2} r_{max}^2 \\
&< \frac{1 - \left(\frac{1}{\rho}\right)^{2(k+1)}}{1 - \left(\frac{1}{\rho}\right)^2} \left(\rho\, c_d(p^*)\right)^2 \\
&\in \mathrm{O}\big(c_d(p^*)^2\big)\,.
\end{aligned}$$

With the equivalence of cost metrics—including the Euclidean metric—in the $\Omega(1)$-model, this holds for any metric. Asymptotic optimality follows from the lower bound example in Chapter 6.

If s and t are not connected, GOAFR$^+$ detects so (case c) in proof of Lemma 7.4) and reports back to the source using the same algorithm. □

Remark Like OAFR and GOAFR, GOAFR$^+$ detects and reports disconnectivity of s and t with cost at most $\mathrm{O}(n' \log n')$, where n' is the number of nodes in the network component containing s.

Having presented different geographic routing algorithms and proved their asymptotic worst-case properties, we will discuss and compare the performance of these algorithms and of other heuristic combinations of greedy and face routing in average-case networks in the following chapter.

Chapter 8

Greedy and Face Routing in Average-Case Networks

It is even harder for the average ape
to believe that he has descended from man.
Henry Louis Mencken (1880–1956)

In the previous three chapters we showed that it is not only possible to define a worst-case-optimal geographic routing algorithm exploiting the concept of face routing, but also to combine the notions of greedy routing and face routing in algorithms which conserve worst-case optimality. The approach to combine greedy and face routing was thereby driven by the observation that—when moving our focus away from worst-case networks to average-case scenarios—greedy routing lends itself as an efficient routing concept. In this chapter we analyze the validity of the hypothesis that a combination with greedy routing can improve performance in average-case networks.

In particular we describe performance measurements taken in simulations not only of the algorithms described in the previous chapters but of a variety of face routing algorithms and their combinations with the greedy approach. We also present a number of graph characteristics yielding deeper insight in the behavior of the routing algorithms. Before focusing on our routing algorithms, we will present a basic observation.

8.1 The Role of Network Density

In this section we will discuss the correspondence between network density and substantial network properties in the context of routing. In particular we point out the importance of the chosen density range for the simulation of routing algorithms.

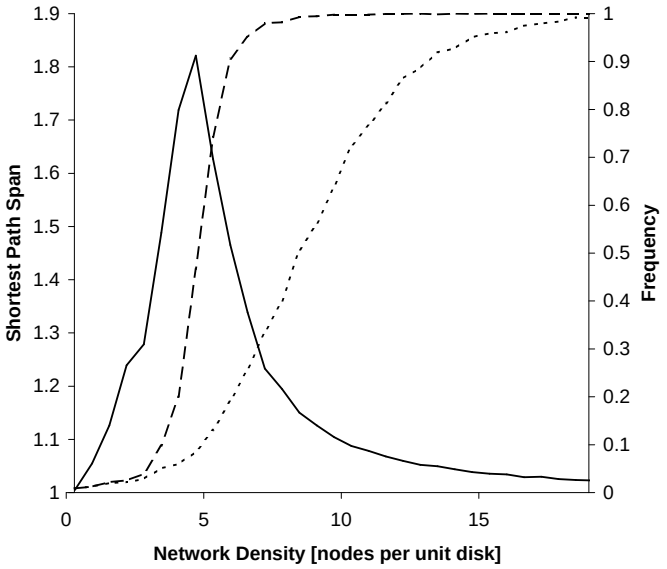

Figure 8.1: Connectivity rate (dashed), greedy success rate (dotted, both plotted against the right y-axis), and mean shortest path span (solid, plotted against the left y-axis).

Our measurements were carried out on the unit disk graph of networks with nodes randomly and uniformly placed on a square with 20 units side length. For each density value (denoted in nodes per unit disk on the x-axis of Figure 8.1), we generated 2000 such random networks and chose the source s and the destination t randomly.

In particular we measured three parameters for each density:

- *Connectivity rate*: In how many cases are s and t connected?
- *Greedy success rate*: How often does the greedy algorithm GR alone reach t?
- *Shortest path span*: The ratio between $c_d(p^*)$, the Euclidean length of the shortest (Euclidean) path, and $|\overline{st}|$, the Euclidean distance between s and t. Note that the shortest path span is only defined if s and t are connected.

Figure 8.1 depicts our measured values of these three parameters over a density range of 0.3 to 20 nodes per unit disk. The connectivity and greedy success rate values are plotted against the right; the mean of the measured shortest path span values is plotted against the left y-axis.

The network connectivity curve (dashed line) shows that the density spans from one extreme to the other. At very low network densities, nodes are so sparsely placed that we observe almost no connectivity at all. On the other end of the scale, s and t are virtually always connected. The transition between these two extremes takes place in a relatively narrow density range between approximately 3 and 7 nodes per unit disk. Network connectivity—above all in this transition range—is one of the main issues in percolation theory [29, 73]. In the following we will justify why this is a mandatory range for routing algorithm simulations to take place.

Of high importance for our greedy/face routing combinations is the greedy success rate (dotted line). Since network connectivity is a requirement for any routing algorithm, the greedy success rate lies strictly below the connectivity rate curve. For high network densities on the other hand, a gap big enough to form a local minimum for greedy routing will only be generated with low probability; greedy routing can be expected to almost always reach the destination.

The third curve in Figure 8.1 (solid line) represents the mean shortest path span, that is, for each density the curve value is the mean of the ratio values between the Euclidean length of the shortest path and the Euclidean distance between s and t over all generated networks. For very low densities this value is close to 1 due to low network connectivity: The shortest path span is only defined if s and t are connected, which is rarely ever the case here. If however s and t are connected, they are with high probability close together or even neighbors, in which case the shortest path span is equal to 1. The low values for high densities, on the other hand, can be explained by the fact that with increasing density the shortest path will more and more closely follow the direct connecting line $\overline{st}$ due to the rising number of nodes in $\overline{st}$'s vicinity.

Between these two extremes the shortest path span forms an almost bell-shaped curve in the network density range approximately between 3 and 6 nodes per unit disk. Informally put, this is the only density region where the shortest path is usually much longer than the direct connection between s and t (cf. Figure 8.2). This specific quality identifies this region as "critical" for routing algorithms. Here, finding a good path at low cost becomes a nontrivial task and a real challenge for geographic routing. It also appears that for the critical density region the effect of introducing "artificiality" by placing network nodes *uniformly* remains acceptably low: The density is relatively heterogeneous, which reflects the situation to be expected in reality more closely than a homogeneous placement of nodes that would occur for higher network densities. Furthermore, it is not astonishing that simulations carried out on dense networks (such as for the GPSR geographic routing algorithm with approximately 21.8 nodes per unit disk in [57]) display very good results with respect to both the quality of the discovered path and algorithm performance.

In the following we will therefore mainly focus on this critical density range around 4.5 nodes per unit disk in our algorithm simulations.

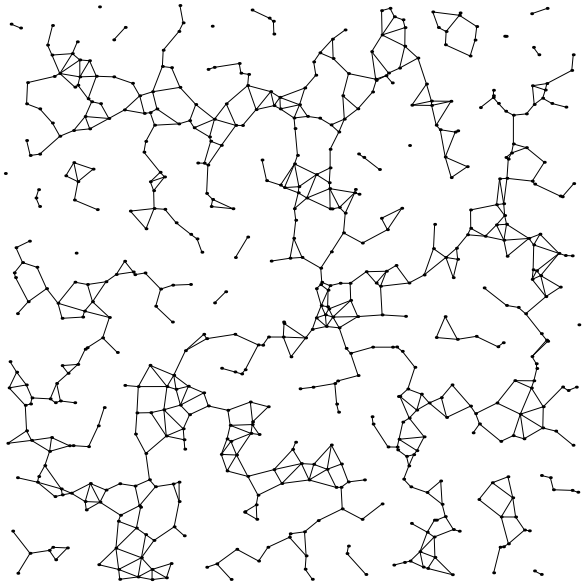

Figure 8.2: Example of generated (Gabriel) graph at critical network density 4.71 ($\approx 1.5\,\pi$) nodes per unit disk (600 nodes on 20 by 20 units field). Most of the nodes are connected; for most pairs of nodes, however, the shortest connecting path is significantly longer than their Euclidean distance.

8.2 Algorithm Overview

Before presenting our simulation results we will introduce all simulated algorithms for the sake of clarity. Figure 8.3 contains an "algorithm family tree", a graphical representation of the conceptual relations between the single algorithms.

The basis of this algorithm family is formed by two fundamental algorithms:

- *FR* is the traditional Face Routing algorithm as introduced in [61] and outlined in Section 5.1. On a planar graph one face after the other is completely explored in order to find intersections of its boundary with $\overline{st}$, the line connecting s and t. Along this line the algorithm gradually proceeds and finally reaches t.

- *GR* proceeds in each step to the neighbor closest to the destination, as described in Chapter 4. Note that this algorithm—as opposed to all other simulated algorithms—cannot guarantee to reach t.

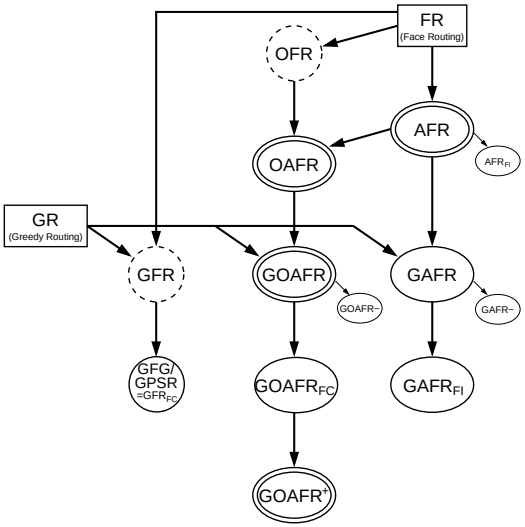

Figure 8.3: Algorithm relation overview. The basis for all simulated routing algorithms is formed by FR and GR. Algorithms in elliptic shapes use a bounding ellipse (GOAFR$^+$ using a bounding circle). A double ellipse represents asymptotic optimality. Algorithms in dashed circles have been introduced for the concept only, without simulations.

In Section 5.3 we introduced OFR, a variant of FR in the sense that it does not switch to a new face along the line $\overline{st}$, but at the point closest to t on the boundary of the currently explored face. OFR builds the basis for a complete line of algorithms; it has however not been implemented for our simulations. Extending the FR and OFR algorithms by adaptive bounding ellipses, we obtain AFR and OAFR:

- In *AFR* (cf. Section 5.2), Face Routing is extended by the concept of a bounding ellipse whose size is iteratively incremented as required. Note that—as an improvement over the original description—our implementation of the AFR algorithm does not return to the source before incrementing the ellipse size.

- In *OAFR*, as described in Section 5.3, face routing is not performed along the $\overline{st}$ line. Instead, the algorithm explores a face for the node on its boundary with the least distance from t, where it proceeds to the next face.

Introduced as a building block for later use, the algorithm AFR_{FI} uses a heuristic to try to switch to the next face earlier than AFR:

- AFR_{FI} (AFR + "First Intersection" heuristic)—in contrast to AFR—does not necessarily explore complete faces but already switches to the next face when *first* meeting an intersection with $\overline{st}$ closer to t than where the current face's exploration started.

The remaining algorithms all include greedy routing phases. GOAFR combines GR with OAFR; GAFR employs GR and AFR. The GFR algorithm is introduced for the concept only, that is without simulations.

- The *GOAFR* algorithm is described in detail in Section 7.1. It employs GR as long as possible and overcomes potential local minima by exploration of one face with OAFR.

- *GAFR* combines greedy routing and AFR. Local minima are circumvented using the AFR algorithm for one face before returning to greedy mode. In analogy to GOAFR, GAFR retains the same ellipse throughout its execution, apart from enlarging it when necessary.

The algorithms GOAFR– and GAFR– are minor variations of GOAFR and GAFR, respectively.

- *GOAFR–*'s only difference compared to GOAFR consists in its use of a bounding ellipse exclusively when in face routing mode. Each time starting a face routing phase at node u, the ellipse is initialized around u and t.

- *GAFR–* reinitializes its bounding ellipse for each face routing phase around the new starting point and t; otherwise it is identical to GAFR.

Similarly as in AFR_{FI}, the algorithms $GAFR_{FI}$, $GOAFR_{FC}$, and GFG/GPSR employ heuristics to terminate the exploration of the current face and consequently fall back to greedy mode earlier than GAFR, GOAFR, or GFR, respectively.

- $GAFR_{FI}$'s only difference compared to GAFR is its use of $\mathrm{AFR_{FI}}$ instead of AFR: The next greedy phase is started already when meeting the first intersection with $\overline{st}$.

- $GOAFR_{FC}$ (GOAFR + "First Closer" heuristic) uses GR and OAFR, but falls back to greedy mode even earlier, that is at the first node closer to t than where the face routing phase started.

- *GFG/GPSR* was introduced in [14] and [57]. Apart from the fact that it does not bound its searchable area, it is identical to $\mathrm{GOAFR_{FC}}$.

Finally, the $\mathrm{GOAFR^+}$ algorithm also tries to return from face routing to greedy routing as early as possible, goes however beyond $\mathrm{GOAFR_{FC}}$ in that it employs a counter mechanism.

Character	Stands for	Comment
R	Routing	occurs in all algorithm names
F	Face	algorithms employing face routing
G	Greedy	algorithms employing greedy routing
A	Adaptive	algorithms using a bounding ellipse
O	Other	algorithms using OFR
–		starts face routing phases with small ellipse
...R_{FI}	First Intersection	uses "First Intersection" heuristic
...R_{FC}	First Closer	uses "First Closer" heuristic
+		early fallback with counters; uses circle with gradually reduced radius (only used in GOAFR$^+$)

Table 8.1: Algorithm naming system.

- *GOAFR$^+$* as described in detail in Section 7.2 applies a node counter mechanism in order to decide when to fall back to greedy routing and thus preserves GOAFR's asymptotic worst-case optimality. Furthermore—in contrast to the other algorithms using an ellipse—it restricts its searchable area with a circle centered at t whose size is gradually reduced while approaching the destination.

In order to distinguish the single algorithms, we introduced a naming system in which each character contained in an algorithm name represents a concept (Table 8.1). Note that the abbreviation GFG/GPSR (Greedy-Face-Greedy/Greedy Perimeter Stateless Routing) [14, 57] does not correspond with our naming system. In our system GFG/GPSR would be named GFR_{FC}.

Table 8.2 summarizes all simulated algorithms and compares them with respect to five characteristics:

- *Type:* The algorithms fall into three main classes: Pure face routing algorithms (fr), the GR algorithm (gr), and combinations thereof (gr + fr).

- *With Bounding Ellipse:* Apart from FR and GFG/GPSR all algorithms bound their searchable area (by an ellipse or, in the case of GOAFR$^+$, by a circle).

- *With Heuristic:* In face routing mode (G)AFR_{FI}, $GOAFR_{FC}$, and GFG/-GPSR apply heuristics in order to proceed with greedy routing without exploring the complete boundary of a face. Doing so, however, these algorithms lose their asymptotic optimality.

- *Retains Ellipse:* This property is only applicable to gr + fr algorithms with a bounding ellipse. Most of these algorithms only use a bounding

ellipse during their greedy routing phase. GOAFR and GAFR in contrast retain—apart from incrementing its size—the same ellipse throughout their complete execution.

- *Asymptotically Optimal:* As outlined in Section 5.2, AFR has previously been proved to be asymptotically optimal. In the subsequent two sections we showed that also OAFR and GOAFR are asymptotically optimal geographic routing algorithms. Note the correspondence between algorithms using heuristics and asymptotic optimality. Only by means of a node counter scheme can the GOAFR$^+$ algorithm preserve worst-case optimality while attempting to fall back to greedy routing without traversing the complete boundary of the currently visited face.

Note that GR is the only algorithm that may fail to reach t. All other simulated algorithms are guaranteed to find the destination.

8.3 Routing Algorithm Simulations

We are now ready to present the results of our routing algorithm simulations. As in the measurements presented in Section 8.1, we generated networks on square fields of side length 20 units by distributing network nodes randomly and uniformly. For every simulation series, the number of nodes was determined according to the chosen network density. For each considered network, the source s and the destination t were also chosen randomly.

In order to judge the practicability of an algorithm we introduced the *normalized cost* $cost_A(N, s, t)$ of an algorithm A in a network N given a source s and a destination t as

$$cost_A(N, s, t) := \frac{s_A(N, s, t)}{c_\ell(p^*_\ell(N, s, t))},$$

where $s_A(N, s, t)$ is the number of steps taken by algorithm A in network N finding a route from s to t (which is in our case, with all simulated algorithms, equal to the number of sent messages); $c_\ell(p^*_\ell(N, s, t))$ is the (hop) length of the shortest path (with respect to the hop metric) between the source s and the destination t in N.

Counting the steps taken by an algorithm A corresponds to the link (or hop) metric of A's path. There are two reasons for choosing the link metric for our simulations. First, the link metric is a model for currently employed radio network technology: In most communication standards (such as IEEE 802.11), radio devices transmit with a fixed—at least not dynamically adapted—power level. Second—assuming that sending a message to a neighboring node takes a certain period of time independent of its distance—the link length of a path forms a good measure for the latency of a message traveling along that path.

Our objective was to measure the performance of the routing algorithms without any interference of possible side effects of other communication layers.

Algorithm Name	Type	With Bounding Ellipse	With Heuristic	Retains Ellipse	Asymptotically Optimal	Comment
FR	fr	no	no	-	no	Face Routing as described in Section 5.1
AFR	fr	yes	no	-	yes	Adaptive Face Routing (Section 5.2)
AFR_{FI}	fr	yes	yes	-	no	AFR, but switches to next face at first intersection with $\overline{st}$
OAFR	fr	yes	no	-	yes	Other Adaptive Face Routing (Section 5.3)
GOAFR	gr + fr	yes	no	yes	yes	Greedy Other Adaptive Face Routing (Section 7.1)
GOAFR–	gr + fr	yes	no	no	no	GOAFR, but starts face routing phases with small ellipse
GAFR	gr + fr	yes	no	yes	no	Greedy + AFR
GAFR–	gr + fr	yes	no	no	no	Greedy + AFR, but starts face routing phases with small ellipse
$GAFR_{FI}$	gr + fr	yes	yes	no	no	GAFR, but falls back to greedy routing when first intersecting $\overline{st}$
$GOAFR_{FC}$	gr + fr	yes	yes	no	no	GOAFR, but falls back to greedy routing when meeting first node closer to t than where AFR phase started
GFG/GPSR	gr + fr	no	yes	-	no	Greedy-Face-Greedy/Greedy Perimeter Stateless Routing [14, 57]: $GOAFR_{FC}$ without bounding ellipse
$GOAFR^+$	gr + fr	yes	-*	-*	yes	$GOAFR^+$ algorithm as defined in Section 7.2. *Uses counters for early fallback and concentric circles
GR	gr	no	-	-	-	out of competition, since reaching of t not guaranteed

Table 8.2: Classification of simulated routing algorithms. The GOAFR algorithm, for instance, is a greedy/face routing combination, employs a bounding ellipse, does not use a heuristic for early fallback to greedy mode, keeps the ellipse when switching from one mode to the other (does not restart with a small ellipse), and is asymptotically optimal. The GR algorithm is listed for completeness, but runs out of competition, since it does not guarantee to reach the destination.

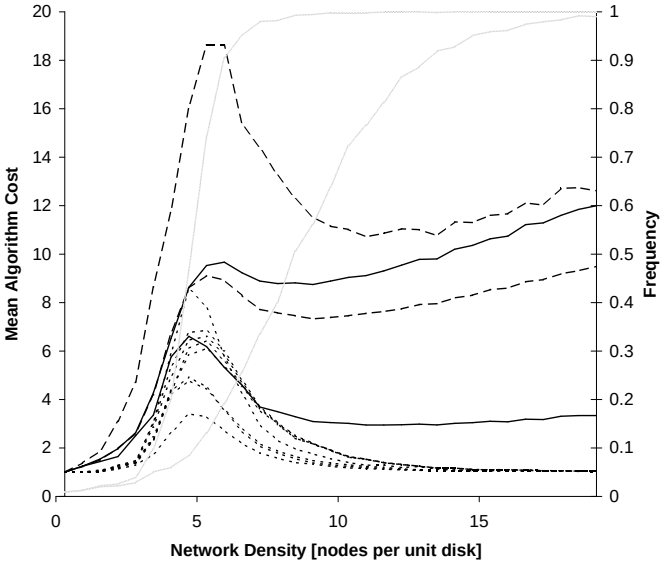

Figure 8.4: Algorithm performance overview. Mean cost values of FR (upper dashed line), AFR (upper solid), OAFR (lower dashed), and AFR_{FI} (lower solid). All greedy/face routing combinations (including GOAFR, GOAFR^+, and GFG/GPSR) are shown in dotted lines (for details see Figure 8.5). The network connectivity and greedy success rates (gray) are plotted against the right y-axis for reference (cf. Section 8.1).

We therefore assumed an ideal environment with collisionless MAC layer and postulated all position information required by the geographic routing model—a node's own and its direct neighbors' positions as well as information about the destination position being present at the source—to be available without additional communication overhead. We furthermore assumed the routing algorithms to execute fast compared to possibly moving network nodes; node movement was consequently not simulated. The measurements were carried out on a custom simulation environment.

Figure 8.4 shows for each simulated algorithm A its mean cost value, defined as

$$\overline{cost_A} := \frac{1}{k} \sum_{i=1}^{k} cost_A(N_i, s_i, t_i)$$

over all $k = 2000$ generated triples (N_i, s_i, t_i) for network densities ranging from 0.3 to 20 nodes per unit disk. For reference to the density range, the network connectivity and greedy success rates are plotted against the right y-axis.

For very low network densities all algorithms perform more or less equally well, for the same reason that keeps the shortest path span low (see Section 8.1): The source s and the destination t are rarely ever connected; if however they happen to be connected, they are very likely direct neighbors.

On the other end of the density scale, two classes of algorithms can be distinguished.

- The cost values of all algorithms solely employing a variant of face routing approach a linearly growing curve. The general growth of the cost of these algorithms towards infinity can be explained by the fact that we measure the hop metric cost values of the algorithm paths together with two reasons: FR, AFR, and OAFR route along complete faces; the expected number of such faces between s and t rises linearly with network density. Although $\mathrm{AFR}_{\mathrm{FI}}$, on the other hand, does not explore complete faces and will—still for dense networks—stay close to the connecting line $\overline{st}$, its cost increases towards infinity since it proceeds along the Gabriel Graph, whose mean edge length decreases with rising density.

- All algorithms combining greedy routing with face routing display cost values approaching 1. For high network densities these algorithms rarely ever need to change to face routing mode (cf. the greedy success rate curve). Furthermore the length of the greedy path approaches the length of the shortest path. Note that in greedy mode all network edges (not only those of the Gabriel Graph) can be used.

The eye-catching bell-shaped cost curves for all algorithms—including the greedy/face routing combinations—are centered around the critical density region as defined in Section 8.1. We observe that—not only in the worst, but also in the average case—FR is clearly disqualified due to its missing limitation to a searchable area; AFR and OAFR—both employing this technique—display much more favorable values, yet cannot compete with $\mathrm{AFR}_{\mathrm{FI}}$, which—at least in the critical region—performs not worse than a number of greedy/face routing combinations.

In accordance with the considerations from Section 8.1, this critical density range appears to be the most challenging area for all the simulated algorithms without exception. Accordingly we will now "zoom" into this area to analyze the performance results of the algorithms combining greedy and face routing in more detail.

Figure 8.5 depicts the mean cost values of all simulated greedy/face routing combinations in the critical density range around 4.5 nodes per unit disk. The eight algorithms can be split into four groups with respect to their peak cost values:

- GOAFR, GOAFR–, GAFR, GAFR–,

- $\mathrm{GAFR}_{\mathrm{FI}}$, $\mathrm{GOAFR}_{\mathrm{FC}}$,

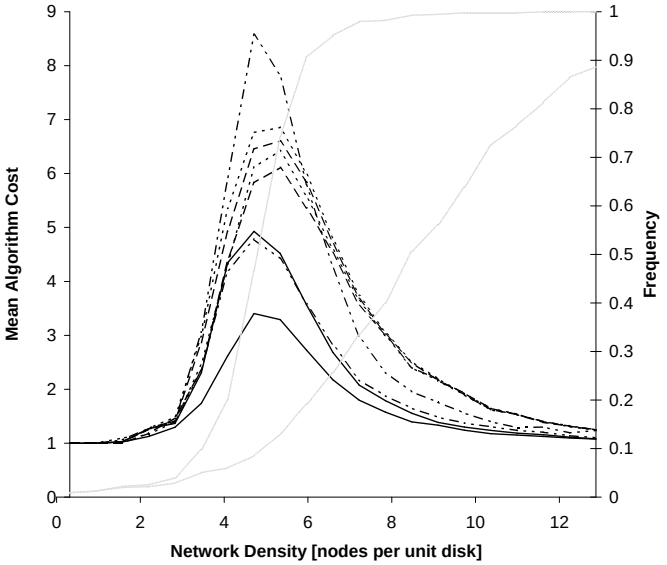

Figure 8.5: Algorithm performance in critical density range around 4.5 nodes per unit disk. Mean cost values of GFG/GPSR (upper dash-dotted line), GOAFR (upper dotted), GAFR (upper dashed), GOAFR– (lower dotted), GAFR– (lower dashed), GOAFR_{FC} (upper solid), GAFR_{FI} (lower dash-dotted), and GOAFR^{+} (lower solid). Again, the network connectivity and greedy success rates are plotted against the right y-axis for reference.

- GFG/GPSR, and

- GOAFR^{+}.

GOAFR, GOAFR–, GAFR, and GAFR– have more or less comparable peak cost values. They have in common that they explore complete faces (within the searchable area) when in face routing mode. It appears that restarting with a small (reinitialized) ellipse at the beginning of each face routing phase slightly reduces mean cost; doing so, GOAFR however loses its asymptotic optimality.

With the heuristics employed by GAFR_{FI} and GOAFR_{FC}, on the other hand, the peaks of the mean cost values are clearly lower. However, in contrast to GOAFR, these two algorithms cannot guarantee asymptotic optimality (cf. Figure 8.6).

GFG/GPSR only differs from GOAFR_{FC} by the absence of a bounded searchable area. Yet this has a dramatic effect on the critical area algorithm cost: While GOAFR_{FC} displays comparatively low cost values, omission of the

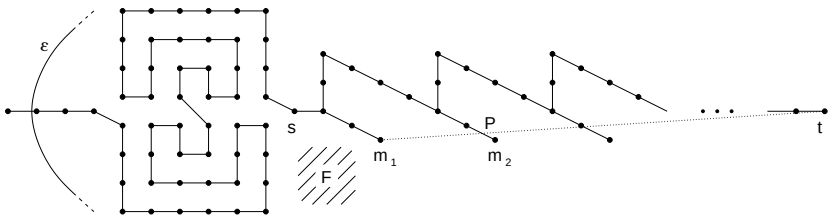

Figure 8.6: Example graph on which (G)AFR$_{\mathrm{FI}}$ and GOAFR$_{\mathrm{FC}}$ display asymptotic suboptimality. Starting from s, GAFR$_{\mathrm{FI}}$, for instance, will reach the local minimum m_1 in greedy mode, switch to face routing mode and (unluckily) begin to explore the boundary of face F in counterclockwise direction. Only after traversing the maze-like structure left of s will the algorithm hit the ellipse $\mathcal{E}$, return (after again passing s) to m_1, and continue to find P, the first intersection with $\overline{m_1 t}$. After falling back to greedy mode and reaching the local minimum m_2, the algorithm will repeat the above procedure. In total, the maze-like structure left of s will be traversed $\Theta(\ell)$ times, where ℓ is the (Euclidean) distance between s and t. (The size of the maze can be chosen in a way that even after $k \in \Theta(\ell)$ of the above rounds it is contained in the ellipse with foci m_k and t.) Since the maze-like structure is designed such that its traversal takes cost $\Theta(\ell^2)$, the overall algorithm executes with cost $\Theta(\ell^3)$, whereas the optimal path has cost $\Theta(\ell)$. Similar examples can be constructed for all simulated algorithms which use heuristics, including GFG/GPSR.

bounding ellipse almost doubles the peak mean cost value, throwing GFG/GPSR back to the last position among the simulated greedy/face routing combinations.

Finally, in order to restrict its searchable area, GOAFR$^+$ employs a circle centered at t with gradually reduced radius while approaching the destination. Mainly owing to this improvement—together with its early-fallback technique—the GOAFR$^+$ algorithm clearly outperforms all other simulated algorithms in average-case networks. This behavior can be observed not only around the critical node density, where this gap is most obvious, but over the complete considered density range. At the same time the GOAFR$^+$ algorithm is, as proved in Section 7.1, worst-case optimal in contrast to all other simulated greedy/face routing combinations (with the exception of GOAFR).

Figure 8.7 displays the (normalized) cost value distributions for all algorithms simulated at network density 4.71 nodes per unit disk. Apart from AFR$_{\mathrm{FI}}$—which also features a significantly smaller mean value at that density (cf. Figure 8.4)—all pure face routing variants show their distribution peaks for relatively high cost values (3–5 for FR, 5–7 for AFR and AFR$_{\mathrm{FI}}$). Due to the lack of a bounding ellipse, FR displays a considerable number of very high cost values (> 41), which is also reflected in its higher mean value at that density.

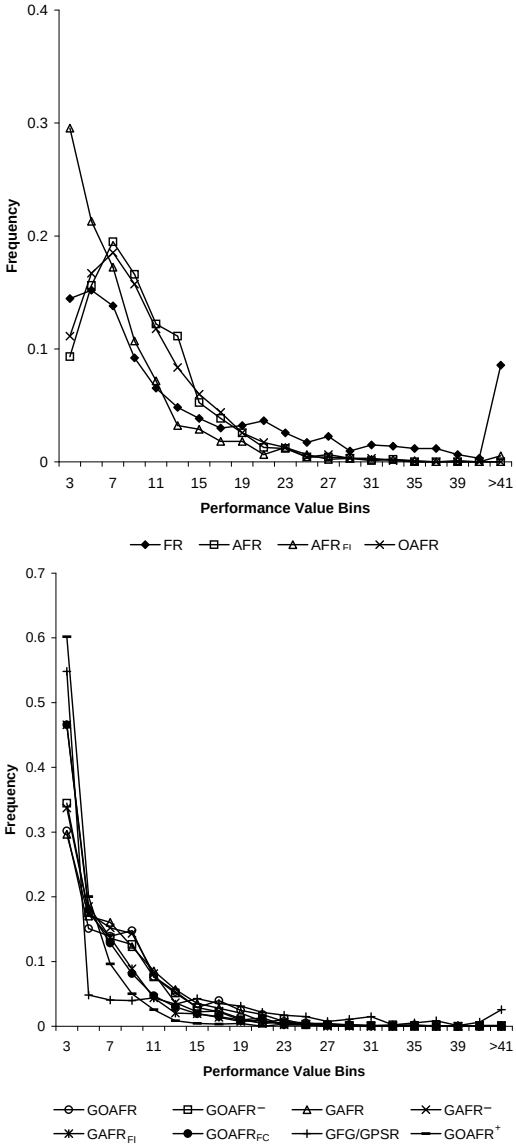

Figure 8.7: Distribution of cost values of the pure face routing algorithms (top) and the greedy/face routing combinations (bottom) simulated at network density 4.71 nodes per unit disk. The simulations for GOAFR$^+$, for instance, show that in approximately 10 percent of all connected graphs cost values greater than 5 but not greater than 7 were measured.

With all greedy/face routing combinations, on the other hand, the majority of the measured cost values are relatively low. Above all GOAFR$^+$ shows efficient behavior, in over 60 percent of all cases reaching the destination with cost values of at most 3. In about half of all simulated networks also the GFG/GPSR, GOAFR$_{\text{FC}}$, and GAFR$_{\text{FI}}$ algorithms have cost at most 3. With the GOAFR–, GAFR–, GOAFR, and GAFR algorithms, finally, roughly half of the cases result in cost values smaller than 5. GFG/GPSR, although featuring a significant number of low cost values, does not restrict its searchable area, which leads to a non-negligible number of large cost values and which consequently also considerably increases its mean value (cf. Figure 8.5).

Remark (Ellipse Size Implementation) For completeness we carried out a simulation series dedicated to the question whether the ellipse size doubling strategy suggested by GOAFR (and AFR)—although compliant with asymptotic optimality—can be improved for practical purposes. We achieved best values initializing the major axis c of the ellipse $\mathcal{E}$ to $1.2 \cdot |\overline{st}|$ and multiplying c with the factor $\sqrt{2}$ (that is doubling $\mathcal{E}$'s area) when $\mathcal{E}$ is required to be enlarged. Employing a circle centered at t for the searchable area, all respective algorithms consistently displayed worse results than using an ellipse. An exception to this observation is formed by GOAFR$^+$, which does benefit from using a circle; the reason for this behavior can be found in GOAFR$^+$—in contrast to all other algorithms—reducing the size of its restricting area whenever possible while approaching the destination. The implementations used in the above simulations were adapted to employ the best parameters found.

8.4 Algorithm Scalability

In the simulation series of Figures 8.4 and 8.5 we analyzed algorithm performance over different network densities, but on a fixed network (field) size. For the following series we considered algorithm performance on different network sizes at fixed network density 4.71 nodes per unit disk.

Figure 8.8 shows the mean performance results obtained in simulations on networks generated in square fields of side length 4 to 40 units. Again, the algorithms fall into different classes with respect to their performance behavior.

The lack of a bounded searchable area results in a fast-growing curve with increasing network size for FR and GFG/GPSR, although on a lower level for the latter algorithm. The most important factor for this behavior is formed by the fact that it can be expected that—for the critical network density—these algorithms are required to explore a considerable part of the entire network independent of its size. If GFG/GPSR can compete with most other algorithms for small networks (up to approximately 12 units side length), this effect clearly disqualifies the algorithm for large networks.

Although GOAFR$_{\text{FC}}$ only differs from GFG/GPSR by using a bounding ellipse, we find this algorithm almost at the other end of the performance scale

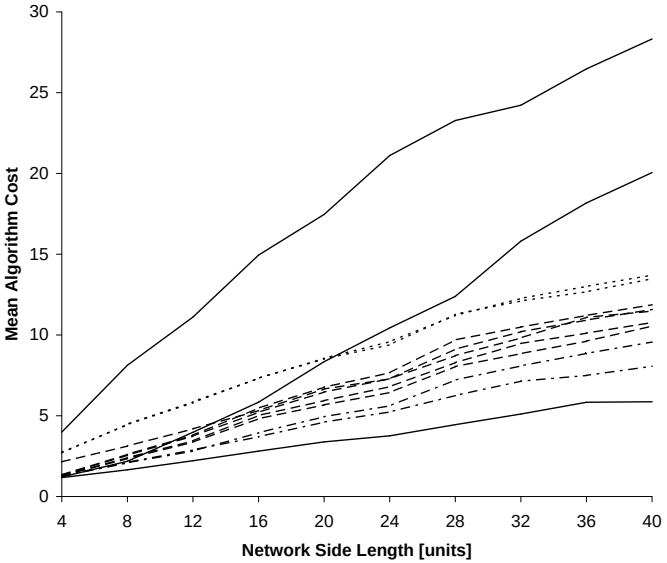

Figure 8.8: Algorithm performance at network density 4.71 nodes per unit disk simulated on networks in square fields with side lengths from 4 to 40 units. Mean cost values of FR (upper solid line), GFG/GPSR (middle solid), AFR (upper dotted), OAFR (lower dotted), GOAFR, GAFR, AFR_{FI}, GOAFR–, GAFR– (dashed lines from top to bottom), GOAFR_{FC} (upper dash-dotted), GAFR_{FI} (lower dash-dotted), and GOAFR^{+} (lower solid).

together with GAFR_{FI}, whose cost values grow relatively slowly for all simulated network sizes.

The results of these two algorithms are only topped by GOAFR^{+}, which appears to be the most resilient algorithm with respect to network growth. As our simulations show, this algorithm not only performs well over a wide node density range, as discussed in the previous sections, but also with increasing network size, where the measured cost values stay below 6 even for the largest simulated networks.

The remaining algorithms all display more or less comparable cost curves, AFR and OAFR—both requiring exploration of complete faces (within the searchable area) and missing a greedy phase—at a slightly higher level. Interestingly AFR_{FI} can compete with GOAFR(–) and GAFR(–); the advantage gained using the "First Intersection" heuristic appears to be neutralized by the lack of a greedy phase (cf. AFR_{FI}).

The general cost growth for increasing network sizes of all algorithms, particularly including the ones using bounding ellipses, can be explained by the

fact that the mean distance (both Euclidean and shortest path) between randomly chosen s and t rises as well and that consequently the expected number of network nodes contained in the searchable area grows even faster.

Chapter 9

On Cost Metrics

What some people mistake for the high cost of living
is really the cost of high living.
Doug Larson

In this chapter we will discuss the properties of cost metrics defined according to Definition 3.2 in the context of geographic routing. In Chapter 3 we proved that all possible thus defined cost metrics are asymptotically equivalent in the $\Omega(1)$-model. In this chapter we will first show that this equivalence of all such cost metrics also holds on unit disk graphs having their node degrees bounded from above by a constant. It follows that the geographic routing algorithms presented in Chapter 7 remain asymptotically optimal also if the $\Omega(1)$-model is replaced by the assumption that the given network graph is a bounded degree unit disk graph.

This property will however not just be stated as a result per se, but also as a fact to be used later in the chapter. In particular we will prove in a second part of the chapter that considering general unit disk graphs—without bounded degree—the cost functions are divided into two classes: *linearly bounded* and *super-linear*. We will show that employing a backbone construction with bounded degree the optimality of the algorithms discussed in Chapter 7 can be extended to general unit disk graphs for linearly bounded cost functions. With super-linear cost metrics on the other hand, a lower bound graph proves that there exists no geographic routing algorithm whose cost is bounded with respect to the cost of a shortest path.

9.1 Bounded Degree Unit Disk Graphs

In Chapter 3 we showed that in the $\Omega(1)$-model the cost of a given path with respect to any two cost metrics differ at most by constant factors. In the

following lemma we will show that a similar property holds in bounded degree unit disk graphs.

Lemma 9.1. *Let $c_1(\cdot)$ and $c_2(\cdot)$ be cost functions according to Definition 3.2 and let G be a bounded degree unit disk graph with node set V and maximum node degree Δ. Further let p be a path from $s \in V$ to $t \in V$ on G such that no node occurs more than a constant k times in p. We then have*

$$c_1(p) \leq \alpha c_2(p) + \beta$$

for two constants α and β. For symmetry reasons, $c_1(p) \in \Theta(c_2(p))$ follows.

Proof. Let $c_d(x) := x$ be the cost function of the Euclidean distance metric. We will show that for any cost function c there exist constants α_1, β_1, α_2, and β_2 such that

$$c(p) \leq \alpha_1 c_d(p) + \beta_1 \text{ and} \tag{9.1}$$

$$c(p) \geq \alpha_2 c_d(p) + \beta_2. \tag{9.2}$$

This means that all cost functions are in $\Theta(c_d(p))$ and particularly $c_1(p) \in \Theta(c_d(p))$ and $c_2(p) \in \Theta(c_d(p))$, which proves the lemma.

We start with Inequality (9.1). Let $c_\ell(x) :\equiv 1$ be the cost function of the link distance metric. Now pick a node u from the path p. As u has at most Δ neighbors and every node is visited at most k times, we leave the disk with radius 1 around u after at most $k\Delta + 1$ steps when starting at u and walking along p. Therefore, the total Euclidean distance of any $k\Delta+1$ consecutive edges of p is at least 1. We then have

$$c_\ell(p) < (k\Delta + 1)\lceil c_d(p)\rceil < (k\Delta + 1)c_d(p) + k\Delta + 1.$$

Because cost functions are nondecreasing, we have $c(e) \leq c(1)$ for any edge e and any cost function $c(\cdot)$. Therefore we get

$$c(p) < c(1) \cdot c_\ell(p) \leq c(1)(k\Delta + 1)\left(c_d(p) + k\Delta + 1\right),$$

which proves Inequality (9.1). Note that as soon as the cost function $c(\cdot)$ is fixed, $c(1)$ is a constant since we required $c(x)$ to be defined for all $x \in]0, 1]$. In order to obtain Inequality (9.2) we will observe that a path p' of length $c_d(p') \geq 1$ has at least one edge e' of length $c_d(e') \geq 1/(k\Delta+1)$: If p' consists of $m < k\Delta + 1$ edges, the longest edge of p' has at least length $1/m$; if p' consists of $k\Delta + 1$ or more edges, we use the fact that $k\Delta + 1$ consecutive edges of p have a total Euclidean length of at least 1. We now partition p into maximal consecutive subpaths of Euclidean length at most 2. All but the last of these subpaths have Euclidean length at least 1, and it therefore follows that

$$\begin{aligned} c(p) &\geq c\left(\frac{1}{k\Delta+1}\right) \cdot \left\lfloor \frac{c_d(p)}{2} \right\rfloor \\ &> c\left(\frac{1}{k\Delta+1}\right) \cdot \left(\frac{c_d(p)}{2} - 1\right), \end{aligned}$$

which concludes the proof. ∎

Note that the geographic routing algorithms presented in Chapter 7 traverse every edge in the graph at most a constant number of times. If the given graph is of bounded degree, also every node is visited at most a constant number of times. Therefore, Lemma 9.1 also applies to the cost expended by a routing algorithm during its execution.

In analogy to the $\Omega(1)$-model, Lemma 9.1 can be applied to the cost of optimal paths.

Lemma 9.2. *Let G be a bounded degree unit disk graph with node set V. Further let $s \in V$ and $t \in V$ be two nodes and let p_1^* and p_2^* be optimal paths from s to t on G with respect to the metrics induced by the cost functions $c_1(\cdot)$ and $c_2(\cdot)$, respectively. We then have*

$$c_1(p_2^*) \in \Theta(c_1(p_1^*)),$$

that is, the cost of optimal paths for different metrics only differ by multiplicative and additive constants.

Proof. This proof is analogous to the proof of Lemma 3.2. By the optimality of p_2^*, we have

$$c_2(p_2^*) \leq c_2(p_1^*). \tag{9.3}$$

p_1^* and p_2^* are cycle-free, and we can apply Lemma 9.1 with $k = 1$. We obtain

$$c_1(p_2^*) \in \Theta(c_2(p_2^*)) \text{ and } c_2(p_1^*) \in \Theta(c_1(p_1^*)). \tag{9.4}$$

Combining Equations (9.3) and (9.4) yields $c_1(p_2^*) \in \mathrm{O}(c_1(p_1^*))$. According to the optimality of p_1^*, we have $c_1(p_2^*) \geq c_1(p_1^*)$; consequently, $c_1(p_2^*) \in \Theta(c_1(p_1^*))$ holds. □

An important property employed in the asymptotic optimality proofs in Chapters 5 and 7 is the fact that in the $\Omega(1)$-model an ellipse (or a circle) can contain at most a number of nodes linear in the size of its area. With the following lemma—showing a similar property for bounded degree unit disk graphs—the optimality proofs also hold if the routing algorithms are executed on unit disk graphs with bounded node degree.

Lemma 9.3. *Let $R \subset \mathbb{R}^2$ be a two-dimensional convex region with area $A(R)$ and perimeter $p(R)$. Further, let V be a set of nodes inside R. If the unit disk graph of V is a bounded degree unit disk graph with parameter Δ (all degrees are at most Δ), the number of points in V is bounded by*

$$|V| \leq (\Delta + 1)\frac{8}{\pi}\left(A(R) + p(R) + \pi\right).$$

Proof. In order to prove this lemma, we first consider the disks with diameter 1. All nodes inside such a disk are less than 1 apart and are therefore adjacent in the unit disk graph. Since the number of neighbors of each node is bounded

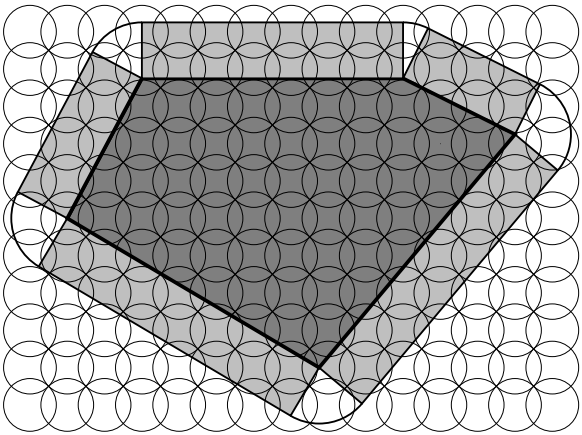

Figure 9.1: Covering a convex region with a grid of equally sized disks.

by Δ, each disk with diameter 1 contains at most $\Delta + 1$ nodes. In order to give a bound on the number of nodes inside the region R, we therefore have to find an upper bound on the number of disks with diameter 1 required to completely cover R. We can cover the whole plane with disks of diameter 1 by placing the disks on an orthogonal grid such that the horizontal and the vertical distances between the centers of two neighboring disks are $1/\sqrt{2}$ (see Figure 9.1). By counting the number of disks intersecting R, we obtain a bound on the number of disks needed to cover R. We see that all disks intersecting R are completely inside the region R', where R' is defined as the locus of all points whose distances from R are at most 1; in other words, we add a border of width 1 to R. Let A' be the area covered by R'. The number of disjoint disks with diameter 1 which can be placed inside R' is bounded by $4A'/\pi$, as the area of a disk with diameter 1 is $\pi/4$. Since in the above defined grid of disks no point in $\mathbb{R}^2$ is covered by more than 2 disks, the number of disks required to cover R can be bounded by $8A'/\pi$. Thus, the number of nodes in V is at most $(\Delta + 1)8A'/\pi$.

In order to obtain the area A', it is sufficient to consider the case where R is a convex polygon. The general case then follows by limit considerations. We get A' by adding $A(R)$, the area of R, and the area of the border around R. As illustrated in Figure 9.1, the border can be broken down into rectangles and sectors of circles. For each side of the polygon R we obtain a rectangle of width 1; and since all the angles of the sectors add up to 2π, the sectors add up to a disk of radius 1. For A' we therefore obtain $A' = A(R) + p(R) + \pi$ where $p(R)$ denotes the perimeter of R. This concludes the proof. □

A smaller constant than $8/\pi$ can be obtained by placing the disks on a hexagonal grid and considering the portion of the area which is only covered by a single disk.

9.2 General Unit Disk Graphs

In this section we will consider the problem of geographic ad-hoc routing on general unit disk graphs (that is of unbounded degree). As shown in the following, the behavior around 0 divides the cost functions defined according to Definition 3.2 into two natural classes. The cost functions lower-bounded by a linear function are called **linearly bounded cost functions**; the cost functions which are not bounded from below by a linear function are called **super-linear cost functions**.

$$\begin{aligned} \text{linearly bounded:} \quad & \exists\, m > 0 :\ c(d) \geq m \cdot d,\ \forall\, d \in]0, 1], \\ \text{super-linear:} \quad & \nexists\, m > 0 :\ c(d) \geq m \cdot d,\ \forall\, d \in]0, 1]. \end{aligned}$$

Of the standard cost measures the link distance and the Euclidean metric are linearly bounded, whereas the energy metric is super-linear. The lower bound example which will be presented in Section 9.2.2 exploits the property that with super-linear cost functions it is possible to construct chains with nodes of distance approaching zero which allow to cover a finite Euclidean distance "for free" in the limit.

We will now describe a geographic routing algorithm which is asymptotically optimal for any *linearly bounded* cost function. We subsequently show that there is no geographic ad hoc routing algorithm whose cost is comparable to the cost of an optimal path for *super-linear* cost functions.

9.2.1 Linearly Bounded Cost Functions

First we will describe our geographic routing algorithm as it can be applied to an arbitrary unit disk graph G and for all linearly bounded cost metrics. In a precomputation phase, a routing backbone G_{BG} is calculated. G_{BG} is a subgraph of G such that a) G_{BG} is a bounded degree unit disk graph and b) the nodes of G_{BG} form a connected dominating set of G.[1] Consequently, all nodes of G have at least one neighbor in G_{BG}. The distributed construction of a subgraph of G with properties a) and b) is described in a number of publications (for example [3, 40, 107]). Such a construction can be achieved for instance by having the nodes elect themselves to be non-neighboring dominators on a first-come-first-serve basis and then connecting these dominators in a second

[1] Given a graph $G = (V, E)$, a node set $D \subseteq V$ is a dominating set if every node in V is either in D or has a direct neighbor in D. A dominating set D is connected if every pair of nodes in D can be connected by a path only using nodes in D.

phase by bridges, each containing at most two intermediate nodes added to the dominating set.[2]

As the backbone contains a dominating set of the underlying graph, every regular node (a node not in the backbone) can be associated to one of its dominators. Since this can be regarded as a clustering of all regular nodes around their dominators, we call this graph the *Clustered Backbone Graph* G_{CBG}. In order to route a message from a regular node s to a regular node t, the message will first be sent to s's associated dominator and then routed along the Backbone Graph to t's associated dominator before finally being forwarded to t itself. Note that while the Backbone Graph is bounded in degree, this is not the case for the Clustered Backbone Graph since a dominator can dominate arbitrarily many other nodes.

The following lemma shows that the length of a route over the backbone is comparable with the optimal route in the given graph for the link metric.

Lemma 9.4. *The Clustered Backbone Graph is a constant-stretch spanner with respect to the link metric, that is, a best path between two nodes on the Clustered Backbone Graph is longer than a path between the same nodes in the underlying unit disk graph by a constant factor only.*

Proof. [107, Lemma 5] proves that the hop length of the shortest path between any two nodes using the Backbone Graph is at most three times (plus an additive constant 2) the hop length of the shortest path in the original unit disk graph, which implies the lemma. □

This property of the Clustered Backbone Graph does not only hold for the link distance metric but for all linearly bounded cost functions.

Lemma 9.5. *The Clustered Backbone Graph G_{CBG} is an asymptotic constant-stretch spanner with respect to any linearly bounded cost metric $c(\cdot)$, that is, the cost of an optimal path on G_{CBG} is only by a constant factor and an additive constant greater than the cost of an optimal path on the underlying unit disk graph G.*

Proof. Let $c_\ell(\cdot)$ be the link distance metric. By Lemma 9.4 we have a path p'_ℓ on G_{CBG} such that $c_\ell(p'_\ell) \in \Theta(c_\ell(p^*_\ell))$, where p^*_ℓ is an optimal link distance path on G. Let p^* denote an optimal path with respect to the cost $c(\cdot)$ on G. We then have to show that $c(p'_\ell) \in \mathrm{O}(c(p^*))$. The Euclidean length of p^* is $c_d(p^*)$ where $c_d(\cdot)$ denotes the cost function of the Euclidean distance metric. We partition p^* into maximal subpaths of length at most 1. Because two consecutive such subpaths have a total length greater than 1, we get at most $\lceil 2\, c_d(p^*) \rceil$ subpaths. We define the path p' by replacing each subpath with a direct edge. Note that all edges of p' have length at most 1. The link distance cost $c_\ell(p')$ of p' is upper-bounded by $c_\ell(p') \leq 2c_d(p^*) + 1$. By the optimality of p^*_ℓ, we also have

[2] The main concepts of this algorithm will be explained in more detail in Section 10.3.

$c_\ell(p') \geq c_\ell(p^*_\ell) \in \Theta(c_\ell(p'_\ell))$. And with respect to the metric $c(\cdot)$, each edge of p'_ℓ has cost at most $c(1)$, leading to $c(p'_\ell) \leq c(1)c_\ell(p'_\ell)$. Together, we get

$$c(p'_\ell) \in \mathrm{O}(c_d(p^*))\,. \tag{9.5}$$

Again note that $c(1)$ is a constant because $c(x)$ has to be defined for all $x \in]0, 1]$. Since $c(\cdot)$ is a linearly bounded cost function, we have $c(x) \geq m \cdot c_d(x)$ for a constant $m > 0$. Therefore also $c(p^*) \geq m \cdot c_d(p^*)$ holds. Combining this with Equation (9.5), we obtain

$$c(p'_\ell) \in \mathrm{O}(c(p^*))\,.$$

□

The routing algorithms described in Chapter 7 operate on planar graphs. As discussed in Chapter 3, we employ the Gabriel Graph to obtain a planar subgraph of the unit disk graph. We will now show that the Gabriel Graph has all required properties to provide for asymptotically optimal routing. In particular, as also shown in Chapter 3, $\mathrm{GG} \cap \mathrm{UDG}$ contains an energy-optimal path, which is the basis of the following lemma.

Lemma 9.6. *Let G be a bounded degree unit disk graph with node set V and let G_{GG} be the intersection of G and the Gabriel Graph of V. Further, we fix two nodes $s \in V$ and $t \in V$. Let $c(\cdot)$ be a cost function and p^* and p^*_{GG} be optimal paths with respect to the metric $c(\cdot)$ on G and on G_{GG}, respectively. We then have*

$$c(p^*_{\mathrm{GG}}) \in \Theta(c(p^*)),$$

that is, G_{GG} is a constant-stretch spanner for all cost functions.

Proof. G_{GG} contains an optimal path with respect to the metric corresponding to the cost function $c(d) := d^2$. Applying Lemma 9.2, we see that the cost of the optimal energy path p^*_{E} is asymptotically equivalent for all cost functions $c(\cdot)$, that is, $c(p^*_{\mathrm{E}}) \in \Theta(c(p^*))$. □

With these observations we can now apply the GOAFR and GOAFR$^+$ algorithms on general unit disk graphs. In a precomputation phase, the Clustered Backbone Graph and its intersection with the Gabriel Graph (on the nodes of G_{CBG}) are constructed. Then routing from the source s to the destination t works as follows.

- If s and t are neighbors in G (the unit disk graph), the message is directly sent from s to t; otherwise, s sends the message to one of its dominators if s is not a dominator itself.

- Then we use GOAFR$^+$ (or GOAFR) to route the message along the Gabriel Graph edges of the Clustered Backbone Graph. As soon as we arrive at a node whose Euclidean distance to t is at most one, the message is directly sent to t. Note that there must be such a node on the boundary of one of the faces we visit.

This application of GOAFR($^+$) on the Clustered Backbone Graph is asymptotically optimal for all linearly bounded cost metrics:

Theorem 9.7. *Let the cost of the best path between a given source-destination pair with respect to a given linearly bounded cost metric be c. The cost of GOAFR($^+$) as described above with respect to the same metric then is $O(c^2)$. This is asymptotically optimal among all possible geographic ad-hoc routing algorithms for linearly bounded cost metrics.*

Proof. Correctness for the case where s and t are direct neighbors follows from the fact that the cost function is linearly bounded. For the other cases we use that the intersection of the Gabriel Graph (on the nodes of G_{CBG}) and the Clustered Backbone Graph is a constant-stretch spanner for linearly bounded cost functions (Lemmas 9.5 and 9.6) and that GOAFR($^+$) expends at most the required worst-case cost on all bounded degree unit disk graphs (Chapter 7 and Section 9.1). Optimality follows, as the $\Omega(c^2)$ lower bound graph from Chapter 6 is also a Clustered Backbone Graph. □

If the GOAFR and GOAFR$^+$ algorithms are asymptotically optimal on general unit disk graphs for linearly bounded cost metrics, the following section will show that this is not true for super-linear cost functions.

9.2.2 Super-Linear Cost Functions

In the remainder of this chapter we will consider geographic ad hoc routing on general unit disk graphs for super-linear cost functions. Unlike for linearly bounded cost functions, the cost of a geographic ad hoc routing algorithm cannot be bounded by the cost of an optimal path in this case.

Theorem 9.8. *Let the best route with respect to a super-linear cost function $c(\cdot)$ for a given source-destination pair be p^*. There is no (deterministic or randomized) geographic ad hoc routing algorithm whose cost is bounded by a function of $c(p^*)$.*

Proof. We construct a family of unit disk graphs in the following way (see Figure 9.2). We choose a positive integer n and place $n+1$ nodes on a straight (say horizontal) line such that two neighboring nodes have distance $0 < d < 1$. Starting with the first node, we mark every $\lfloor 2/d \rfloor^{\text{th}}$ node. For every marked node u_i we then place a node v_i such that $\overline{u_i v_i}$ has length 1 and such that all the new nodes lie on a line which is parallel to the line on which we put the first $n+1$ nodes. This yields k vertical edges of length one. The distance between two such edges is $D = \lfloor 2/d \rfloor d$. Note that $1 < D \leq 2$, as we have chosen d to be smaller than 1. The number of marked nodes (that is the number of vertical edges) k is then bounded by

$$k = \left\lfloor \frac{dn}{D} \right\rfloor \geq \left\lfloor \frac{dn}{2} \right\rfloor > \frac{dn}{2} - 1. \tag{9.6}$$

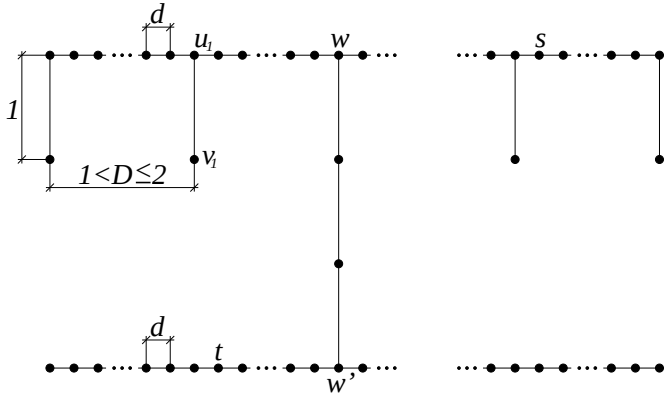

Figure 9.2: Lower bound graph for super-linear cost functions.

Now we choose an arbitrary marked node (we call it w) and the corresponding v_i. At v_i we add two other vertical edges and arrive at node w' which has distance 3 from the line with the original $n+1$ nodes. Symmetrically to the original $n+1$ nodes, we now place another row of $n+1$ nodes (including w') on a horizontal line with distance 3. Figure 9.2 illustrates this construction. We choose an arbitrary node of the top $n+1$ nodes for the source s. The destination t is also chosen arbitrarily from the bottom $n+1$ nodes. The optimal route p^* from s to t then first goes from s to w, then from w to w', and finally from w' to t. The cost of p^* can be bounded by $c(p^*) \leq 2nc(d) + 3c(1)$.

We want this cost to be constant and therefore choose $c(d) = 1/n$, yielding $d = c^{-1}(1/n)$. Note that since $c(\cdot)$ has to be nondecreasing, $c^{-1}(\cdot)$ is well-defined as long as there are no intervals where $c(\cdot)$ is constant. For those intervals we define $c^{-1}(\cdot)$ to take any of the possible values. For the cost of the optimal path $c(p^*)$ we now obtain a constant value ($c(1)$ is a constant!), that is, $c(p^*) \in \Theta(1)$. In order to compute the cost of a geographic ad hoc routing algorithm $\mathcal{A}$, we observe that $\mathcal{A}$ has no information about the location of w and therefore has to test all possible nodes by using the k edges of length 1. For a deterministic $\mathcal{A}$ we can always place w such that it is the last marked node which is tried. For a randomized $\mathcal{A}$ we can place w such that the expected number of required trials is at least $k/2$. For the cost $c(\mathcal{A})$ of any geographic ad hoc routing algorithm, we therefore obtain $c(\mathcal{A}) \in \Omega(k)c(1) = \Omega(k)$. Plugging $d = c^{-1}(1/n)$ into Equation (9.6), we have

$$k > \frac{1}{2}nc^{-1}(1/n) - 1.$$

For n approaching infinity we then obtain

$$\begin{aligned}\lim_{n\to\infty} k &\geq \lim_{n\to\infty} \frac{1}{2} n c^{-1}(1/n) - 1 \\ &= \frac{1}{2} \lim_{y\to 0} \frac{c^{-1}(y)}{y} - 1 \\ &= \frac{1}{2} \lim_{x\to 0} \frac{x}{c(x)} - 1 \;=\; \infty,\end{aligned}$$

where we substituted $y := 1/n$ in the first step and $x := c^{-1}(y)$ in the second step. The last limit is ∞ by the definition of $c(\cdot)$, a super-linear cost function, which implies that $\lim_{x\to 0} c(x)/x = 0$ if this limit exists. (For convenience we assume that the limit exists. Otherwise the same result can be achieved by "tuning" the graph more closely to the cost function.) Therefore, the cost of any algorithm $\mathcal{A}$ is unbounded with respect to the cost of an optimal path p^*, which has constant cost. □

Chapter 10

Geographic Routing Beyond Unit Disk Graphs

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.
Albert Einstein (1879–1955)

Up to this point of this dissertation the analysis of the geographic routing algorithms has been based on the assumption that the given communication network is a unit disk graph. As described in Chapter 3, nodes are located in the Euclidean plane and are assumed to have identical (unit) transmission radii. Consequently, an edge between two nodes—representing that they are in mutual transmission range—exists if and only if their Euclidean distance is not greater than one. On the one hand, clearly, this is a glaring simplification of reality, since, even if all network nodes are homogeneous, this model does not account for the presence of obstacles, such as walls, buildings, mountains—or also weather conditions—, which might obstruct signal propagation. On the other hand, unit disk graphs are simple enough to allow of strong theoretical results, such as the routing guarantees discussed in the previous chapters.

In this chapter we will study a graph model, originally introduced in [7], which is considerably closer to reality. We will maintain the assumption that all mobile nodes are placed in the plane (that is, they have coordinates in $\mathbb{R}^2$). In a *quasi unit disk graph*, two nodes are connected by an edge if their distance is less than or equal to d, d being a parameter between 0 and 1. Furthermore, if the distance between two nodes is greater than 1, no edge exists between them. In the range between d and 1, the existence of an edge is not specified.

In this chapter we will first establish—in analogy to Chapter 6—a constructive lower bound for quasi unit disk graphs showing that basically any algorithm without routing tables requires sending of $\Omega\big((\frac{c}{d})^2\big)$ messages to route from a source s to a destination t, where c is the length of the shortest path between s

and t. We will show that, with the aid of a topology control graph structure, a restricted flooding algorithm is guaranteed not to perform worse and that this technique is consequently asymptotically message-optimal.

If we additionally make the basic assumptions of geographic routing—attributing the network nodes with information about their own and their neighbors' positions and assuming that the message source knows the position of the destination—, a more subtle approach than flooding of the network is possible. We will present a combination of greedy routing and restricted flooding. This yields a routing algorithm that is still asymptotically optimal in the worst case, but also efficient in the average case, as suggested by the results on average-case efficiency of geographic ad hoc routing algorithms presented in the previous chapters. Finally we will show that, if we assume d to be at least $1/\sqrt{2}$, it is possible to locally introduce virtual edges and perform the geographic routing algorithms discussed so far while preserving performance guarantees known from unit disk graphs.

In the following section we will specify the models and provide definitions that go beyond Chapter 3. In Section 10.2 we will establish a lower bound for the message complexity of so-called volatile memory routing algorithms. Section 10.3 contains the description of the topology control structure forming the basis for the subsequent algorithms. Section 10.4 provides the analysis of flooding algorithms with respect to message and time complexity. Section 10.5 discusses the combination of flooding with a greedy approach for geographic routing, whereas Section 10.6 shows that for large enough d, geographic routing as presented in the previous chapters can be employed.

10.1 Model

This section provides definitions of the model employed in this chapter where they have not been presented in Chapter 3. We will first give a formal definition of our ad hoc network model:

Definition 10.1. (Quasi Unit Disk Graph) *Let V be a set of nodes in the 2-dimensional plane $\mathbb{R}^2$ and $d \in [0,1]$ be a parameter. The symmetric Euclidean graph (V,E), such that for any pair of nodes $u,v \in V$*

- $(u,v) \in E$ *if* $|uv| \leq d$ *and*
- $(u,v) \notin E$ *if* $|uv| > 1$,

where $|uv|$ is the Euclidean distance between the nodes u and v, is called a quasi unit disk graph (quasi-UDG) with parameter d.

In the subsequent section we will establish a lower bound for the message complexity of so-called volatile memory routing algorithms. With this model, nodes are attributed with a short-term memory in which for each message a constant number of bits may be stored temporarily.

Definition 10.2. (Volatile Memory Routing Algorithm) *The task of a volatile memory routing algorithm is to transmit a message from a source s to a destination t on a graph, where each node of the graph holds a memory in which $\mathrm{O}(\log n)$ bits may be stored as long as the message is en route, where n is the number of network nodes.*

In particular, this model allows the nodes to store message identifiers—having a bit length logarithmic in the number of nodes—required for flooding (cf. Section 10.4).

The second important algorithm model discussed in this chapter is geographic routing according to Definition 3.4. As stated there, in original geographic routing a node is allowed to store messages only temporarily before relaying them. In order to enable an algorithm to employ flooding, this restriction has to be relaxed:

Definition 10.3. (Geographic Volatile Memory Routing Algorithm) *A geographic volatile memory routing algorithm is a volatile memory routing algorithm additionally observing the first three rules of the definition of geographic routing algorithms (cf. Definition 3.4).*

In the previous chapters we discussed routing algorithms sending not more than one message at a time. In this chapter we will consider algorithms using message flooding, where several messages can be sent simultaneously. In order to account for this modified algorithm behavior, we will provide in the following a concise overview of basic concepts of distributed computing vital for the understanding of this chapter. More detailed descriptions can be found in textbooks, such as in [86].

At certain points of the chapter we have to distinguish between the *synchronous* and the *asynchronous* model of distributed computation. In the *synchronous* model, communication delays are assumed to be bounded. As a consequence, it can also be assumed that all processes running on different network nodes perform their message sending and receiving operations in simultaneous and globally clocked rounds. In the *asynchronous* model, message delays are unbounded. No assumptions can be made on the duration of single process operations.

Two fundamental measures in distributed computing are *message* and *time complexity*. The *message complexity* of a distributed algorithm is the total number of messages sent during its execution. The definition of *time complexity* depends on the synchrony model: In the synchronous model, time complexity is the total number of rounds elapsed between algorithm start and algorithm termination. In the asynchronous model, such a simple time model cannot naturally be obtained since the transmission delay of a message is unbounded. The common solution to this is the assumption that the message delay is at most one time unit.

Finally, since we consider message complexities in this chapter, we define the *cost of a path* according to the link distance metric, that is, the cost of a

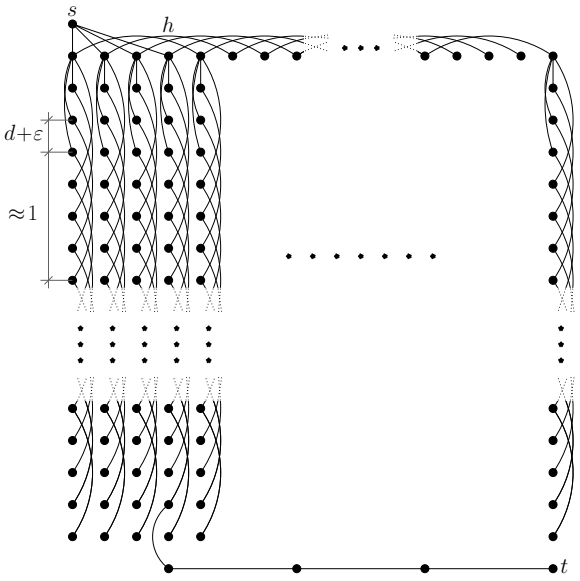

Figure 10.1: Message complexity lower bound for volatile memory routing algorithms on quasi unit disk graphs.

path is the number of edges on the path. Similarly, we consider *spanner* graphs with respect to the link distance metric: A graph $G' = (V, E')$ is a spanner of a graph $G = (V, E)$ with stretch factor k if and only if for any pair of nodes (u, v) the cost of the shortest path on G' is at most k times the cost of the shortest path on G.

10.2 A Lower Bound on Quasi Unit Disk Graphs

Before discussing particular routing algorithms, we will present in this section a lower bound on the message complexity of any volatile memory routing algorithm. In analogy to Theorem 6.1, this is shown constructing a family of graphs.

Theorem 10.1. *Let c be the cost of a shortest path from s to t. There exist graphs on which any (randomized) volatile memory routing algorithm has (expected) message complexity $\Omega((\frac{c}{d})^2)$.*

Proof. We provide a constructive proof by describing a class of graphs for which the theorem holds. The basic element used for the construction of these graphs

is formed by k nodes (k to be determined later) equidistantly placed on a line, such that the distance between two adjacent nodes is $d+\varepsilon$ for a small $\varepsilon > 0$ (cf. vertical chains in Figure 10.1). There exists an edge between every pair of nodes (u, v), such that $(\lceil \frac{1}{d} \rceil - 1)\, d < |uv| \leq 1$, that is, the nodes are connected by all the edges with maximum Euclidean length not greater than 1. In addition there is a head node having an edge to each one of the first $\lceil \frac{1}{d} \rceil - 1$ nodes on the line (the head node is located such that all additional edges have length at most 1). As shown in Figure 10.1, k such vertical chains are placed side by side with distance $d+\varepsilon$ such that the nodes form a matrix. The head nodes of these chains are now interconnected in a way that they among themselves have the same chain structure (uppermost row in Figure 10.1) with their head node (of second order) denoted by s. The node t—located near the bottom right corner of the node matrix—is connected to one of the end nodes of exactly one of the vertical chains by a simple chain of nodes. Note that the constructed graph is a quasi unit disk graph.

The main property posing a problem for a routing algorithm is that a matrix column consists of $\lceil \frac{1}{d} \rceil - 1$ interleaved chains which are only connected via the head node. (The same also holds for the first matrix row.) Consequently only one of the neighbors of s leads to h, the head node of the column connected to t, and only one of the neighbors of h leads to the bottom node connected to t. Since a volatile memory routing algorithm has no a priori information about the graph structure, a deterministic algorithm has to explore every matrix node before finding the path to t. (For a randomized algorithm, t can be connected to the matrix such that the algorithm has to explore roughly half of the matrix nodes in expectation.) A volatile memory routing algorithm therefore has to send $\Omega(n)$ messages, where n is the total number of nodes. The optimal path on the other hand—almost exclusively using edges of length nearly 1—has cost about $2\,k \cdot d$, which—together with $k \approx \sqrt{n}$—establishes the theorem. □

10.3 Topology Control

In the previous section we introduced a lower bound graph class in which any volatile memory routing algorithm cannot find the destination with message complexity less than $\Omega((\frac{c}{d})^2)$. In this section we will now describe how to obtain a subgraph of a given quasi unit disk graph which forms the basis for our algorithms matching the lower bound. Similar to the construction presented in Section 9.2.1, this *Backbone Graph* features two important properties exploited for routing: (1) It contains in a given area A at most $\mathrm{O}\big(\frac{A}{d^2}\big)$ nodes and (2) it is a $\mathrm{O}\big(\log(\frac{1}{d})\big)$-spanner.

Given a quasi unit disk graph G, the Backbone Graph is constructed in three steps. Steps 1 and 2 can be performed by a standard distributed algorithm (as already mentioned in Section 9.2.1) by having the nodes send `dominator` and `connector` messages. The details of this algorithm are omitted, as such discussion would go beyond the scope of this dissertation.

1. The first step consists of a clustering process. We construct a Maximal Independent Set *MIS* of the nodes in G. Note that since *MIS* is an independent set in G, any two nodes in *MIS* have distance greater than d and consequently a given area A contains at most $O(\frac{A}{d^2})$ nodes in *MIS*. For the purpose of routing, the nodes in *MIS* will later become cluster heads: Since the nodes in *MIS* also form a Dominating Set, any node in G will have at least one node from *MIS* within its neighborhood and will choose one of these as its cluster head.

2. In a second step the cluster heads are linked together by connector nodes, connecting all pairs of nodes in *MIS* that are at most three hops apart in G. This results in the *Dense Backbone Graph* G_{DBG}. Since *MIS* is a Dominating Set, the cluster heads can be connected by bridges consisting of at most two nodes. Furthermore, G_{DBG} is a constant-stretch spanner of G.

3. G_{DBG} can contain $\Omega(\frac{A}{d^4})$ nodes in a given area A, which is too many by a factor of $\frac{1}{d^2}$ compared to the lower bound. The size of *MIS* matching the lower bound, the third step now reduces the number of connecting bridges between cluster heads. Let $G^{(v)}_{DBG}$ denote the graph with node set *MIS* and (virtual) edges between all nodes connected by bridges in G_{DBG}. Our objective is now to construct a subgraph $G^{(v)}_{BG}$ of $G^{(v)}_{DBG}$ with $O(\frac{A}{d^2})$ (virtual) edges within the area A. It eventually follows that the final Backbone Graph G_{BG}—where the (virtual) edges in $G^{(v)}_{BG}$ have again been replaced by connector nodes and their adjacent edges—contains at most $O(\frac{A}{d^2})$ nodes within the area A.[1]

   In order to obtain a graph $G^{(v)}_{BG}$ with the desired property, the plane is divided by a grid into square cells of side length 6. In each cell z all nodes and edges completely contained within z temporarily form a local network. (Note that we assume for this operation that the nodes are informed about their positions.) The number of nodes contained within z is at most $O(\frac{1}{d^2})$. We now apply an algorithm constructing a sparse spanner [4, 86, 87] to reduce the number of edges contained in z to $O(\frac{1}{d^2})$.[2] This procedure is repeated three times on grids with their origins shifted by $(3,0), (0,3)$, and $(3,3)$, respectively, relative to the origin of the first grid (cf. Figure 10.2). Note that these are local operations since the subgraphs are of bounded size. The edge set of graph $G^{(v)}_{BG}$ is finally formed by the union of all edges resulting from the edge reduction steps on all four grids.

[1] Note that this graph G_{BG} is not identical to the Backbone Graph denoted with the same symbol in Section 9.2.1.

[2] The mentioned algorithm constructs for a constant $\kappa \geq 1$ an $O(\kappa)$-spanner with at most $n^{1+1/\kappa}$ edges. Setting $\kappa = \log n$ and since $n = \frac{1}{d^2}$, we obtain a graph with the required properties.

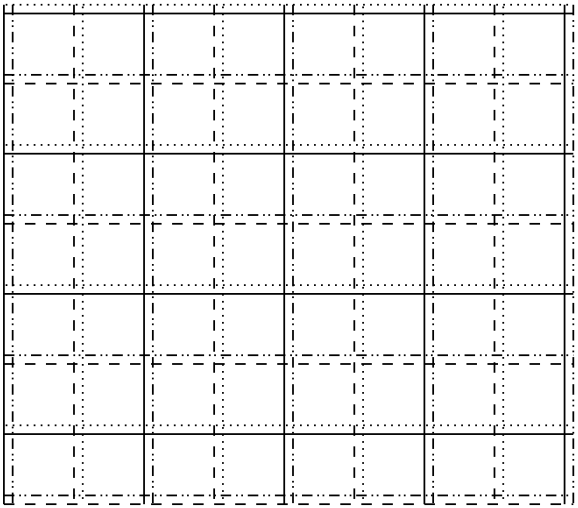

Figure 10.2: Grid structure employed in the construction of a sparse spanner.

The following lemma now proves the first essential property of this subgraph of the given quasi unit disk graph.

Lemma 10.2. *In a given area A (with constant extension in each direction), the number of nodes and the number of edges in the Backbone Graph are both bounded by* $\mathrm{O}\left(\frac{A}{d^2}\right)$.

Proof. The grids employed for the edge reduction steps are chosen to have two properties: (1) Every edge in $G_{DBG}^{(v)}$ is completely contained in at least one cell and (2) any region (with constant extension in each direction) is intersected by at most a constant number of grid cells (for instance a square of side length 3 can be intersected in total by at most 9 grid cells). Property (1) guarantees that every edge is considered at least with one of the four grids: Together with the fact that the edge reduction step does not alter the number of components in a cell subgraph, it follows that the number of components in the complete graph is not altered either. Since each resulting subgraph contains at most $\mathrm{O}\left(\frac{1}{d^2}\right)$ edges and together with Property (2), it follows that also the union of all remaining edges—that is the number of edges in $G_{BG}^{(v)}$ is not greater than $\mathrm{O}\left(\frac{1}{d^2}\right)$ for a constant region. The fact that each edge in $G_{BG}^{(v)}$ corresponds to at most two nodes and three edges in G_{BG} and $G_{BG}^{(v)}$ having at most $\mathrm{O}\left(\frac{A}{d^2}\right)$ nodes for an area A (the nodes in *MIS*) finally leads to the lemma. □

The second essential property of G_{BG} is shown in the following lemma.

Lemma 10.3. *The Backbone Graph G_{BG} is a spanner of G_{DBG} with stretch factor $\mathrm{O}\big(\log(\frac{1}{d})\big)$.*

Proof. Every edge in $G_{DBG}^{(v)}$ is contained in at least one grid cell and consequently also considered in at least one of the according subgraphs. Since the edges retained in each subgraph form a $\mathrm{O}\big(\log(\frac{1}{d})\big)$-spanner (on the subgraph), this property also holds for the union of all subgraphs, $G_{BG}^{(v)}$. Finally, each edge in $G_{BG}^{(v)}$ resulting in at most three edges in G_{BG}, the lemma follows. □

In distributed computing, a distinction is made between the *one-hop broadcast* model and the *point-to-point communication* model: In the one-hop broadcast model, a node can simultaneously send a message to all its neighbors, whereas in the point-to-point communication model, a message is sent over an edge to one distinct neighbor. The algorithms described in the remaining sections of this chapter are assumed to execute on G_{BG}. Since on this graph the number of nodes and the number of edges are asymptotically equal in a given area, the two models can be employed interchangeably, depending on whether we argue over the number of nodes or edges in the graph.

Similar to the backbone routing process in Section 9.2.1, when routing a message m from a source s' to a destination t', the nodes s' and t' will in general not be cluster heads. The complete process of routing therefore consists of

1. s' sending m to its associated cluster head s,
2. routing m from s to t, the cluster head associated to t', and
3. t sending m to t'.

Since steps 1 and 3 incur only constant cost with respect to both message and time complexity, we exclusively consider step 2 in the remaining part of the chapter. Whenever mentioning a source s or a destination t we therefore assume that s and t are cluster heads.

10.4 Message-Optimal Flooding

In this section we will discuss the message and time complexities of the Echo algorithm on quasi unit disk graphs. For succinctness we only give a short outline of the algorithm execution; more detailed information can be found for instance in [18, 86]. The Echo algorithm consist of a flooding phase and an echo phase.

- The flooding phase is initiated by the source s by sending a flooding message—containing a time-to-live (TTL) counter τ—to all its neighbors. Each node receiving the flooding message for the first time decrements the TTL counter by one and retransmits the message to all its neighbors

(with the exception of the neighbor it received the message from). In the synchronous model, this flooding phase constructs a Breadth First Search (BFS) tree.

- From the leaves of this tree—the nodes where the τ counter reaches 0—echo messages are sent back to the source along the BFS tree constructed during the flooding phase. An inner node in the BFS tree can decide locally when to send an echo message to its parent in the tree by awaiting receipt of echo messages from all of its children.

By initiating the first flooding phase with τ set to 1 and relaunching a flooding phase with doubled τ whenever the echo messages indicate that the destination has not yet been reached, both time and message complexities can be bounded:

Theorem 10.4. *Employed on G_{BG} in the synchronous model, the Echo algorithm reaches the destination with message complexity $\mathrm{O}\big((\frac{c}{d})^2\big)$ and time complexity $\mathrm{O}\big(c \cdot \log(\frac{1}{d})\big)$, where c is the cost of a shortest path between s and t. This is asymptotically optimal with respect to message complexity.*

Proof. The Echo algorithm floods the complete network with message complexity $\mathrm{O}(m)$ and time complexity $\mathrm{O}(D)$, where m is the number of edges in the network and D is the diameter of the network. Since no edge in G_{BG} is longer than 1, all nodes reached with a certain τ lie within the circle centered at s with radius τ. The number of edges within this circle is bounded by $\mathrm{O}\big((\frac{\tau}{d})^2\big)$. Note that the destination is reached at the latest for the maximal τ less than $2 \cdot c$. Since τ is doubled after each failure, the total number of visited edges is formed by a geometric series and consequently asymptotically dominated by the number of edges in the circle with maximum τ, from which the message complexity follows. Asymptotic optimality follows from the lower bound established in Section 10.2.

The time complexity follows from the fact that the BFS tree constructed during the flooding phase contains a shortest path from s to t. Since G_{BG} is a $\log(\frac{1}{d})$-spanner of G, the shortest path on G_{BG}, on which the algorithm is executed, is $c \cdot \log(\frac{1}{d})$. The time complexity of a single flooding-echo round being proportional to τ and again the total time complexity asymptotically being dominated by the maximum τ used, the time complexity follows. □

In the asynchronous model, the synchronizer construction introduced in [5] can be employed.

Theorem 10.5. *When employed on G_{BG} in the asynchronous model, the Echo algorithm reaches the destination with message complexity $\mathrm{O}\big((\frac{c}{d})^2 \cdot \log^3(\frac{c}{d})\big)$ and time complexity $\mathrm{O}\big(c \cdot \log(\frac{1}{d}) \cdot \log^3(\frac{c}{d})\big)$, where c is the cost of the shortest path between s and t.*

Proof. The synchronizer construction introduced in [5] incurs an additional cost factor of $\mathrm{O}\big(\log^3(n)\big)$, where n is the number of involved network nodes, with

respect to both message and time complexity. As in our case the number of involved nodes is in $O\big((\frac{c}{d})^2\big)$, this cost factor is in $O\big(\log^3(\frac{c}{d})\big)$. Plugging in the above Echo algorithm for the synchronous model yields the lemma. □

For geographic routing as discussed in the following section, a variant of the Echo algorithm can be defined by replacing the time-to-live counter by a geometric argument: The flooding message is retransmitted only by nodes located within a circle centered at s with a certain radius r.

Lemma 10.6. *The geographic Echo algorithm reaches t with message and time complexity $O\big((\frac{c}{d})^2\big)$, where c is the link cost of the shortest path. This holds for both the synchronous and the asynchronous model and is asymptotically optimal with respect to message complexity.*

Proof. In contrast to the above Echo algorithm using TTL, all nodes located within the restricting circle centered at s with radius r participate in the execution of the geographic algorithm. This circle containing at most $O\big((\frac{r}{d})^2\big)$ nodes, the message complexity follows, where the remaining reasoning is analogous to the one in the proof of Theorem 10.4. The time complexity follows from the fact that time complexity cannot be greater than message complexity. □

10.5 Greedy Echo Routing

Although asymptotically message-optimal, a flooding-based algorithm is prohibitively expensive in most networks for practical purposes. Chapter 7 showed how this problem can be tackled by combining a correct routing algorithm (which is guaranteed to find the destination) with a greedy routing scheme. In this section we will follow this example by describing a geographic volatile memory routing algorithm that tries to leverage the advantages of a greedy routing approach with respect to both conceptual simplicity and message-efficiency: In order to route a message, a node simply forwards it to its neighbor closest to the destination. As discussed in Chapter 4, greedy routing can however run into a local minimum with respect to the distance to the destination, that is a node without any neighbors closer to t. In the algorithm described below, such a local minimum is overcome by employment of restricted flooding, in particular by the aid of the geographic Echo algorithm as described in the previous section. In this section we will therefore refer by Echo to the geographic Echo algorithm. We denote with Echo_r the subalgorithm of geographic Echo consisting of the flooding and the corresponding echo phases for the radius r.

Similar to the algorithms presented in Chapter 7, our algorithm GEcho combines both greedy routing and flooding in two modes: Generally the message is forwarded in greedy mode as long as possible. Whenever running into a local minimum, the algorithm switches to echo mode. In order to keep the cost of flooding-based echo low, the algorithm tries to fall back to greedy mode as early as possible. The fallback criterion is chosen such that the combined

routing algorithm is asymptotically optimal with respect to message complexity. In particular, the Echo algorithm does not terminate only when finding t, but already when finding a node v which is significantly closer to t than the local minimum, as described in Step 2 of the GEcho algorithm:

GEcho The value q is a constant parameter chosen prior to algorithm execution such that $0 < q \leq 1$.

0. Start at s.
1. **(Greedy Mode)** Forward the message to the neighbor in G closest to t. If t is reached, terminate. If a local minimum is reached, continue with step 2, otherwise repeat step 1 at the next node.
2. **(Echo Mode)** Execute algorithm Echo starting at the local minimum u until either reaching t—in which case the algorithm terminates—or finding a node v, such that $|ut| - |vt| \geq q \cdot r$, where r is the currently chosen radius in Echo_r, the subalgorithm of Echo using radius r. Proceed to v and continue with step 1.

In the following we will obtain a statement on the asymptotic complexity of the algorithm. We will first show that the number of messages sent in greedy mode is bounded:

Lemma 10.7. *The number of messages sent in greedy mode is bounded by* $\mathrm{O}\big((\frac{c}{d})^2\big)$.

Proof. Let us exclusively consider the sequence U of nodes sending messages in greedy mode or receiving messages sent in greedy mode during the execution of the algorithm. Note that the distance to t is strictly decreasing within U. Since the algorithm stays in greedy mode until reaching a local minimum, U is partitioned into subsequences $U_1, U_2, \ldots, U_k, k \geq 1$ of nodes by the occurrence of local minima: A local minimum only receives a greedy message without being able to send it to a subsequent node in greedy mode. Within a subsequence $U_i = u_1, u_2, \ldots, u_{\ell_i}$, $\ell_i \geq 2$, any two nodes u_j, u_{j+2} with $1 \leq j \leq \ell_i - 2$ have distance greater than d (otherwise u_j would have sent the greedy message directly to u_{j+2}). On the other hand also the distance between a local minimum u_{ℓ_i} and the first node in the following subsequence U_{i+1} have distance greater than d (otherwise u_{ℓ_i} would be not a local minimum). Together with the fact that all nodes in U are located within the circle C centered at t with radius $|st|$, the number of nodes in the total sequence U is therefore bounded by twice the maximum number of nodes with relative distance greater than d—or likewise the maximum number of nonintersecting disks of radius $d/2$—that can be placed within C. With $|st| \leq c$, the lemma follows. □

We will now confine ourselves to the number of messages sent in echo mode. Note that after each *round*, defined to be one execution of Step 1 or Step 2, the algorithm is strictly closer to t than before that round.

Lemma 10.8. *For a given r, the subalgorithm* Echo_r *is executed at most* $\lceil \frac{|st|}{q\,r} - 1 \rceil$ *times.*

Proof. According to the criterion described in Step 2, an echo round initiated at node u terminates—unless arriving at t—only if it finds a node v such that $|ut| - |vt| \geq q \cdot r$. For any r (also if at a particular node Echo_r fails and r is doubled) such a progress can be made at most $\lceil \frac{|st|}{q\,r} - 1 \rceil$ times, since after each round the algorithm is strictly closer to t than before. □

With this property we can obtain the total number of messages sent in echo mode during algorithm execution.

Lemma 10.9. *The total number of messages sent in echo mode is at most* $\mathrm{O}\big((\frac{c}{d})^2\big)$.

Proof. We obtain the total number of messages sent in echo mode by summing up over all nodes ever contained in a circle bounding Echo_r. Since the number of nodes contained in a given circular area is asymptotically proportional to the size of this area, it is sufficient to compute the total area covered by all Echo_r bounding circles. Let $r_i = 2^i$, $i = 1, 2, 3, \ldots$ denote the radii of the echo-bounding circles. The maximum r_i can be found by the observation that (1) all echo-restricting circles have their centers at a node not farther from t than s and (2) the circle centered at any node not farther from t than s having radius $2\,c$ completely contains the shortest path. Since the value of r in Echo_r is obtained by doubling, the maximum r_i used overall is less than $4\,c$; the maximum i reached is consequently $\lceil \log(4\,c) \rceil$. With R_i being the total number of bounding circles used with radius r_i, we obtain

$$A = \sum_{i=0}^{\lceil \log(4\,c) \rceil} R_i \cdot \pi r_i^2$$

for the total covered area A. Using Lemma 10.8 we obtain

$$\begin{aligned} A &\leq \pi \cdot \sum_{i=0}^{\lceil \log(4\,c) \rceil} \lceil \frac{|st|}{q\,r_i} - 1 \rceil \cdot r_i^2 \\ &< \pi \cdot \sum_{i=0}^{\lceil \log(4\,c) \rceil} \frac{|st|}{q} \cdot r_i \underset{(|st| \leq c)}{\leq} \frac{\pi\,c}{q} \cdot \sum_{i=0}^{\lceil \log(4\,c) \rceil} 2^i \\ &= \frac{\pi\,c}{q} \cdot \big(2^{\lceil \log(4\,c) \rceil + 1} - 1\big) \;\in\; \mathrm{O}\big(c^2\big)\,. \end{aligned}$$

The area A containing at most $\mathrm{O}\big(\frac{A}{d^2}\big)$ nodes (cf. Section 10.3), the lemma follows. □

In total, the complexity of the GEcho algorithm can be bounded as follows:

Theorem 10.10. *The algorithm GEcho finds the destination with both message and time complexity* $O((\frac{c}{d})^2)$*, where* c *is the link cost of the shortest path.*

Proof. The message complexity bound follows directly from the previous two lemmas. The time complexity bound follows from the fact that time complexity cannot be greater than message complexity. □

Corollary 10.11. *The algorithm GEcho is asymptotically optimal with respect to message complexity.*

Proof. Follows from Theorem 10.10 and Section 10.2. □

10.6 Large d-Values

This section treats the special case where the parameter d of the quasi unit disk graph G is $d \geq 1/\sqrt{2}$. This case has already been considered by in [7]. It is shown there that for $d \geq 1/\sqrt{2}$, standard geographic routing is possible. Here we extend their results and present a geographic routing algorithm which is asymptotically optimal, that is whose cost is quadratic in the cost of an optimal path (cf. Chapter 6).

The structural difference between quasi-UDGs for $d < 1/\sqrt{2}$ and quasi-UDGs for $d \geq 1/\sqrt{2}$ lies in the local environment of intersecting edges. If $d \geq 1/\sqrt{2}$, all intersections can be detected locally. This is shown by the following two lemmas.

Lemma 10.12. *Let* $e = (u, v)$ *be an edge and* w *be a node which is in the disk with diameter* (u, v)*. Either* u *and* w *or* v *and* w *are connected by an edge.*

Proof. The following proof is illustrated by Figure 10.3. Because $|uv| \leq 1$, the regions of the points whose distances to u and v are greater than $1/\sqrt{2}$ do not intersect inside $\mathcal{C}$ (shaded areas in Figure 10.1). Thus either $|uw| \leq 1/\sqrt{2}$ or $|vw| \leq 1/\sqrt{2}$. In the figure this holds for u and w, implying that G contains an edge between these two nodes. □

Lemma 10.13. *Let* $e_1 = (u_1, v_1)$ *and* $e_2 = (u_2, v_2)$ *be two intersecting edges in a quasi-UDG* G *with parameter* $d \geq 1/\sqrt{2}$*. Then at least one of the edges* (u_1, u_2)*,* (u_1, v_2)*,* (v_1, u_2)*, or* (v_1, v_2) *exists in* G*.*

Proof. We have to show that one of the four sides of the quadrangle (u_1, u_2, v_1, v_2) is shorter than $1/\sqrt{2}$. Because the sum of the interior angles of the quadrangle is 2π, at least one of the angles has to be greater or equal to $\pi/2$. Assuming without loss of generality that this is the angle at node u_2, u_2 lies in the disk with diameter (u_1, v_1), and the lemma follows from Lemma 10.12. □

We will now give an overview of the results of [7]. The algorithm consists of three steps. In a first step, the quasi-UDG G is extended by adding virtual edges. Whenever there is an edge (u, v) and a node w which is inside the circle with

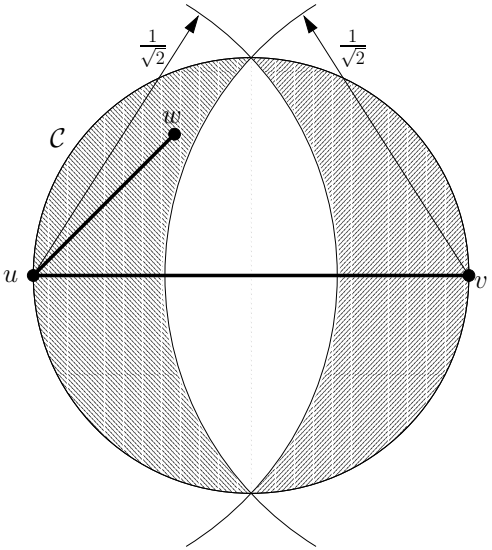

Figure 10.3: For $d \leq 1/\sqrt{2}$, w is either connected to u or to v (cf. proof of Lemma 10.12.

diameter (u, v), for at least one of the nodes u and v—without loss of generality let it be u—the distance to w is smaller than or equal to $1/\sqrt{2}$ (Lemma 10.12), and therefore u has a connection to w. If there is no edge between v and w, a virtual edge is added. Sending a message over this virtual edge is done by sending the message via node u. This process is done recursively, that is, also if (u, v) is a virtual edge. The graph obtained by adding the virtual edges to G is called the super-graph $S(G)$. Barrière et al. prove that on $S(G)$ the Gabriel Graph $GG(S(G))$ can be constructed yielding a planar subgraph of $S(G)$ (cf. Chapter 3). Then any geographic routing algorithm guaranteed to reach the destination is applied on $GG(S(G))$.

In order to obtain an optimal geographic routing algorithm, we have to change the algorithm of [7] in two ways: i) The planar graph which we need for geographic routing should be a constant-stretch spanner and the number of nodes in a given area A should not exceed $O(A)$. ii) We have to replace the geographic routing algorithm by a more elaborate variant such as the algorithms presented in Chapter 7.

One of the bounding factors for the spanning property is given by the recursive depth of the virtual edge construction, that is the length of paths corresponding to virtual edges. From [7] we have the following result.

Lemma 10.14. *Let λ be the minimum Euclidean distance between any two nodes. If $d \geq 1/\sqrt{2}$, the length of the route in G corresponding to a virtual edge in $S(G)$ is at most $1 + \frac{1}{2\lambda^2}$.*

Proof. The lemma follows directly from Property 1 in Section 5 of [7]. □

As shown later in the section, the assumption that there is a minimum Euclidean distance λ between any two nodes would be sufficient to allow for the formulation of asymptotically optimal geographic routing algorithms. However, even without this assumption, but employing the Backbone Graph G_{BG} (cf. Section 10.3), we obtain a quasi-UDG with bounded degree, a property which we will prove to be equivalent to the minimum distance assumption.

Precisely, we start by constructing G_{BG}. This gives us a set of dominator nodes $D = MIS$ and a set of connector nodes C. We transform G_{BG} into a quasi-UDG $G'_{BG} = (V', E')$ by setting $V' = D \cup C$ and by including all possible edges of E in E' (all edges between nodes of V').[3]

Lemma 10.15. *The degree of each node in the quasi-UDG G'_{BG} is bounded by a constant.*

Proof. Because the dominator nodes D have distance at least $1/\sqrt{2}$ from each other, the number of dominators which are within three hops from a node $v \in V'$ is bounded by a constant. Only these dominators can add connector nodes which are neighbors of v. Each of them can only add a constant number of connector nodes; therefore the degree of node v is constant. □

G'_{BG} is now used for the Gabriel Graph construction. First, virtual edges are added as in the algorithm of [7], resulting in a super-graph $S(G'_{BG})$. Then $GG(S(G'_{BG}))$ is constructed. In analogy to Lemma 10.14, we can state a bound on the maximum route length for any virtual edge.

Lemma 10.16. *Let $G = (V, E)$ be a quasi-UDG with maximum node degree Δ. If $d \geq 1/\sqrt{2}$, the length of the route in G corresponding to a virtual edge in $S(G)$ is at most $\mathrm{O}(\Delta^2)$.*

Proof. Let $(u, v) \in E$ be an edge of G. Further let $w_1, \ldots, w_k$ be a sequence of nodes which recursively force the creation of new virtual edges e_i for which the corresponding route contains (u, v). Let $\ell_0 := |uv|$ and ℓ_i be the Euclidean length of the virtual edge e_i (see Figure 10.4 as an explanation). λ_i is the length of the edge which together with e_{i-1} provides the route for e_i ($\lambda_i \leq d$). For the length ℓ_i of the i^{th} virtual edge e_i we obtain

$$\ell_i \leq \sqrt{\ell_{i-1}^2 - \lambda_i^2} \;=\; \sqrt{1 - \frac{\lambda_i^2}{\ell_{i-1}^2}} \cdot \ell_{i-1} \leq \; \sqrt{1 - \lambda_i^2} \cdot \ell_{i-1}.$$

[3]Note that G'_{BG} can contain more edges than G_{DBG} introduced in Section 10.3, as G'_{BG} contains *all* edges between nodes in V'.

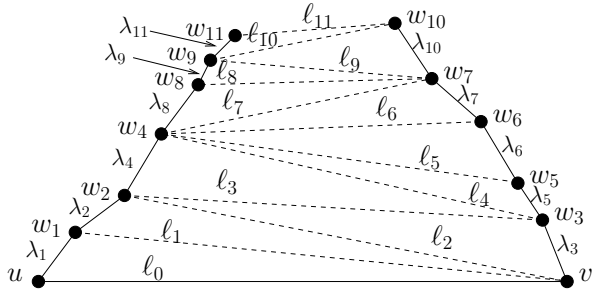

Figure 10.4: Recursive depth of virtual edges in $S(G)$ (cf. proof of Lemma 10.16).

The last inequality follows from $\ell_{i-1} \leq 1$. We therefore have

$$\ell_k \;\leq\; \prod_{i=1}^{k} \sqrt{1-\lambda_i^2} \cdot \ell_0 \;\leq\; \prod_{i=1}^{k} \sqrt{1-\lambda_i^2}. \tag{10.1}$$

We define $\lambda := 1/k \sum_{i=1}^{k} \lambda_i$ to be the average length of the edges corresponding to the λ_i. As we will show in Lemma 10.17, the expression of Equation (10.1) can be upper-bounded by replacing each λ_i by λ:

$$\ell_k \;\leq\; \left(\sqrt{1-\lambda^2}\right)^k \;=\; \left(1-\lambda^2\right)^{k/2}. \tag{10.2}$$

In a quasi-UDG all nodes in a disk with radius $d/2$ are direct neighbors. Therefore, when starting at a node u, after at most $\Delta + 1$ hops, a cycle-free path must leave the disk with radius $d/2$ around u. Thus, the sum of the lengths of $\Delta + 1$ successive edges on a cycle-free path is greater than $d/2$; in particular, for $d \geq 1/\sqrt{2}$ this is a constant. The average edge length of any cycle-free path is thus at least $\Omega(1/\Delta)$. As illustrated in Figure 10.4, the λ_i form two paths. Therefore, the average λ_i must be in the order of $\lambda \in \Omega(1/\Delta)$. As $(1-1/n)^n \leq 1/e$, we can set $k = 2/\lambda^2 \in \mathrm{O}\big(\Delta^2\big)$ in (10.2) and obtain

$$\ell_k \leq (1-\lambda^2)^{1/\lambda^2} \leq 1/e \leq 1/\sqrt{2}.$$

According to the definition of a virtual edge, the length of such an edge e_k is at least $\ell_k \geq 1/\sqrt{2}$. Accordingly, k cannot be chosen greater than in $\mathrm{O}\big(\Delta^2\big)$, which concludes the proof. □

In Equation (10.2) we used that ℓ_k can be upper-bounded by replacing each λ_i by the average edge length λ. This is proved in the following lemma:

Lemma 10.17. *Given k real numbers $\lambda_1, \ldots, \lambda_k$ with $|\lambda_i| \leq 1$,*

$$\prod_{i=1}^{k} \sqrt{1-\lambda_i^2} \leq \left(\sqrt{1-\lambda^2}\right)^k$$

holds, where $\lambda := 1/k \sum_{i=1}^{k} \lambda_i$.

Proof. To prove this lemma, it is sufficient to show that the inequality holds for replacement of two values λ_i and λ_j by their average. It then follows that replacing $\lambda_{min} := \min \lambda_i$ and $\lambda_{max} := \max \lambda_i$ by $\lambda_{avg} := (\lambda_{min} + \lambda_{max})/2$ does not make the product $\prod_{i=1}^{k} \sqrt{1-\lambda_i^2}$ smaller. Repeated application of this substitution of λ_{avg} for λ_{min} and λ_{max} to the newly obtained set of λ_i values results in a chain of inequalities in which λ_{min} and λ_{max} (in every respectively updated set of λ_i values) converge to λ (defined to be the average of all *initial* λ_i values) and at the end of which stands $\ldots \leq \left(\sqrt{1-\lambda^2}\right)^k$.

We will first show that

$$\sqrt{1-\lambda_i^2}\sqrt{1-\lambda_j^2} \leq 1 - \overline{\lambda}_{ij}'^2, \tag{10.3}$$

where $\overline{\lambda}_{ij}' := \sqrt{(\lambda_i^2 + \lambda_j^2)/2}$. In other words—setting $x_i := \lambda_i^2$ and $x_j := \lambda_j^2$—we show that $\sqrt{1-x_i}\sqrt{1-x_j} \leq 1 - (x_i + x_j)/2$. Squaring this equation and subtracting the left hand side from the right hand side, we obtain

$$0 \leq \frac{x_i^2}{4} - \frac{x_i x_j}{2} + \frac{x_j^2}{4} = \frac{(x_i - x_j)^2}{4},$$

which holds for any real x_i and x_j and therefore implies the correctness of Equation (10.3).

Defining $\overline{\lambda}_{ij} := (\lambda_i + \lambda_j)/2$, subtraction of $\overline{\lambda}_{ij}^2$ from $\overline{\lambda}_{ij}'^2$ leads to

$$\overline{\lambda}_{ij}'^2 - \overline{\lambda}_{ij}^2 = \frac{\lambda_i^2}{4} - \frac{\lambda_i \lambda_j}{2} + \frac{\lambda_j^2}{4} = \frac{(\lambda_i - \lambda_j)^2}{4} \geq 0,$$

the last inequality holding again for all real λ_i and λ_j, which implies $\overline{\lambda}_{ij}^2 \leq \overline{\lambda}_{ij}'^2$. Together with Equation 10.3,

$$\sqrt{1-\lambda_i^2}\sqrt{1-\lambda_j^2} \leq 1 - \overline{\lambda}_{ij}'^2 \leq 1 - \overline{\lambda}_{ij}^2$$

holds, which proves that the product $\prod_{i=1}^{k} \sqrt{1-\lambda_i^2}$ does not become smaller after replacement of both λ_i and λ_j by $\overline{\lambda}_{ij}$. Consequently this also holds for λ_{min}, λ_{max}, and λ_{avg}, which—together with the observation made at the beginning of the proof—establishes the lemma. □

Having thus completely proved the correctness of Lemma 10.16, it can now be employed to show that shortest paths are longer on $GG(S(G'_{BG}))$ than on G by at most a constant factor.

Lemma 10.18. *The Gabriel Graph $GG(S(G'_{BG}))$ is a spanner for the quasi-UDG G.*

Proof. From Lemma 10.16 we see that the virtual edges only impose a constant factor on the cost of a path. We can therefore proceed as if all virtual edges were normal edges of G. Further, the Gabriel Graph construction retains an energy-optimal path (see Lemma 3.3). As the average edge length of $S(G'_{BG})$ is constant (cf. proof of Lemma 10.16), the number of hops and the energy cost of a path only differ by a constant factor. Therefore, the minimum energy path is only by a constant factor longer than the shortest path connecting two nodes. Further details are analogous to the considerations in the previous chapters concerning cost metric equivalence on unit disk graphs. □

We can now state the main result of this section.

Theorem 10.19. *Let G be a quasi unit disk graph with $d \geq 1/\sqrt{2}$. Applying OAFR, GOAFR, or GOAFR$^+$ (cf. Chapters 5 and 7) on $GG(S(G'_{BG}))$ yields a geographic routing algorithm whose cost is $\mathrm{O}(c^2)$, where c is the cost of an optimal path. This is asymptotically optimal.*

Proof. As G can be the unit disk graph—setting $d := 1$—, the lower bound follows from the lower bound for unit disk graphs in Chapter 6. The number of nodes as well as the number of edges of $GG(S(G'_{BG}))$ in a given area A is proportional to A; therefore the $\mathrm{O}(c^2)$ cost also directly follows from the respective analyses in Chapters 5 and 7. □

10.6.1 Alternative Construction

We conclude the section on quasi unit disk graphs for $d \geq 1/\sqrt{2}$ with the description of an alternative construction of a planar graph which can be used to perform geographic routing. By Lemma 10.13, all edge intersections of a quasi-UDG with $d \geq 1/\sqrt{2}$ can be detected locally (in one communication round). Instead of the virtual edges/Gabriel Graph construction, we can define virtual nodes at all intersections of two edges. These virtual nodes are managed by the endpoints of the intersecting edges; sending a message from or to a virtual node means sending a message from or to a (non-virtual) neighbor of the virtual node. If this is applied on G'_{BG}, we obtain a planar graph (by definition!) with only $\mathrm{O}(A)$ nodes in any given area A. Because this planar graph is a spanner, we obtain a geographic routing algorithm with cost $\mathrm{O}(c^2)$ by applying AFR, GOAFR, or GOAFR$^+$.

Chapter 11

How to Learn About the Destination Position

A sailor without a destination cannot hope for a favorable wind.
Leon Tec

If you don't know where you are going, any road will take you there.
Lewis Carroll (1832–1898), in 'Alice's Adventures in Wonderland'

The two fundamental assumptions made for geographic routing are (1) that every node is informed about its own and its neighbors' positions and (2) that the source of a message knows the position of the destination.

Depending on the considered scenario, the first assumption is more or less justified. Considering the rescue team scenario, for instance, it is very likely that rescuers are equipped with Global Positioning System (GPS) receivers, as the availability of position information is crucial in rescue and emergency situations regardless of its potential use for geographic routing. In car fleet networks, a second frequently proposed scenario for ad hoc networks, a node's own position information can be considered present almost for free, a currently increasing number of cars being equipped with GPS receivers present as a component of built-in navigation systems. Where employment of GPS is not possible—GPS requires an unobstructed line-of-sight to a considerable portion of the sky—or where dependence on an external system is not desired, a second approach to the acquisition of position information is worth mentioning: There have been numerous proposals for systems which aim at estimating the relative position of nodes based on the presence of certain anchor nodes which are informed about their positions [79, 82, 99]. These approaches employ trilateration—a node determining its position based on distance estimates obtained from received signal strengths or from measured signal arrival time differences—or triangulation—position inference based on measured signal arrival angles. In the absence of

anchor nodes knowing their positions, a similar approach computes "virtual coordinates" of the nodes such that embedding the nodes in the Euclidean plane according to these coordinates results in a network whose neighborhood structure corresponds (as closely as possible) to the topology in the given network graph [9, 76, 96].

In general, it can be stated that there exist important scenarios where the scenario itself implies the presence of information about the position of a network node's own position. In such scenarios information about a node's own position—and, assuming that this information can be exchanged among neighboring nodes, also about the position of neighbors in the network—the first assumption appears to be legitimate.

The second fundamental assumption of geographic routing—the source of a message being informed about the position of the destination—appears to be a more intricate issue. In the following we will discuss a selection of perspectives and approaches with respect to this second assumption.

Again, certain scenarios are conceivable where assuming the presence of knowledge about the destination position appears to be unproblematic. In particular in "geocasting" [60, 81], the destination of a message is not addressed as a specific network node, but as a geographic region; depending on the circumstances, all or a subset of the nodes located in that region are the intended message recipients. In this sense, the second basic assumption of geographic routing holds as a consequence of the chosen scenario.

In most scenarios, however, the destination of a message is required to be addressable as an individual entity. In a generic sense, the task of providing node location information in an ad hoc network is subsumed under the term "location service". Location services in principle offer the two operations of publishing and looking up the location of a given node.

A simple location service could consist in an initial phase—launched if the message source has no or outdated information about the position of the intended destination—where the network is flooded with a request message by having every node receiving that message for the first time rebroadcast it among its neighbors. When the message arrives at the destination, it can reply by sending its current position to the original source (whose position is included in the request message). In order not to unnecessarily flood the whole network, a technique with exponentially increasing time-to-live values—similar to the procedure described in Section 10.4—can be applied. After this initial request-reply cycle, both the source and the destination are mutually informed about their positions. If nodes are moving, position updates can later be piggybacked to messages exchanged between the source and the destination in order to refresh the position information stored at the respective communication partners. In the presence of moving network nodes and provided that a given pair of nodes exchanges messages frequently enough, this piggybacking technique can be employed as an optimization regardless of the initial position lookup approach.

Another location service could be designed along the lines of the solution chosen for Mobile IP [90, 91], where, simply put, communication with a mobile

station takes place via a home agent, which is informed and regularly updated about how and in which physical network the mobile node can currently be reached. Similarly, for every node in an ad hoc network, a fixed physical location can be defined which later acts—for instance adopted by the node currently nearest to that location—as a rendezvous point both for position publication or updates sent by the mobile node and for position lookup requests issued by message sources trying to contact the considered mobile node. As the "home location" is fixed and known in advance, both the publish and lookup operations can employ geographic routing. In contrast to Mobile IP, where due to several reasons not only connection setup but also subsequent communication takes place via the home agent, with geographic routing—once the source is informed about the destination position—actual communication will proceed directly between the source and the destination.

The problem of a fixed home location attributed to a node is that the route distance between the source and the home agent as well as the distance between the home agent and the destination can happen to be much longer than the distance between the source and the destination. In other words, the cost of both the publish and the lookup operations may stand in no relation to the actual distance between the source and the destination. This issue is addressed in the GLS Grid Location System [69]. GLS defines a set of globally known hierarchical grids overlaid on the network area, where every grid cell of a given hierarchy level contains four cells of the next lower hierarchy level. Every node publishes its position to a specific node in each of the three nearest sibling cells on every hierarchy level. This publishing process takes place using geographic routing via intermediate nodes whose positions are known owing to the grid structure. Also as a consequence of this structure, position information is distributed less and less densely with increasing distance from the considered node. The lookup operation takes place in a similar manner as publishing, starting from the source and searching—by means of geographic routing—specific nodes in neighboring sibling cells on increasing hierarchy levels until finding a node informed about the destination position. The main benefit of the Grid Location System is that in many cases the effort expended for the lookup operation corresponds to the distance between the source and the destination.

Bounding the cost of the publish and lookup operations even in worst-case node configurations and motion patterns has been achieved by the Locality-Aware Location Service LLS [1]. Similar to GLS, LLS itself employs geographic routing and uses sets of hierarchical grids. In contrast to GLS, however, every destination is attributed its own set of aligned hierarchical grids. On every hierarchy level, a node stores position pointers at the four corners of the grid cell in which it is located, or more precisely at the node nearest to the respective corner. As in GLS, the density of informed nodes decreases with growing distance from the destination. The lookup operation proceeds in a similar way as the publish operation, by querying the corners of the cells in which the source is located on increasing hierarchy levels until a position pointer to the destination is found; the destination can subsequently be found by following the previously

published position pointers. LLS provides two cost guarantees: First, the cost of the lookup operation is proportional to the distance between the source and the destination. Second, (except for initialization) the cost of the publish operation is proportional to the distance a moving node has covered since its latest publishing.

If node mobility incurs additional cost in the above systems and is generally considered a negative factor, last encounter routing [45] aims at leveraging the effect of node mobility. In particular, it exploits the fact that information can be diffused within the network by moving nodes. Unlike the above location services, last encounter routing does not publish position information by sending messages to specific storage nodes. Instead, every node keeps a local database storing for every other node the place and time it has last encountered that node. The Exponential Age SEarch algorithm EASE is based on this encounter information. Starting at the source—and assuming for succinctness of explanation that the source met the destination some period of time T ago—, EASE searches the nodes around its current position until finding a node v that met the destination significantly later than the source, particularly at most a period of time $T/2$ ago. Using geographic routing, the message proceeds to the position where v has last met the destination, in an optimization of the original algorithm trying to collect newer encounter information on its way there. Once the message reaches a node without any newer destination position information, this procedure is iterated by restarting a search around the current location of the message. Notably, last encounter routing does not incur transmission of information distribution messages but is exclusively based on the information diffusing effect of node mobility. Notably, applicability of the principle of last encounter routing is not restricted to geographic routing, this approach also being practicable in networks where no position information is available [31].

It can be summarized that not only there exist various scenarios whose condition themselves justify the basic assumptions of geographic routing, but that numerous approaches have been proposed and analyzed with the goal of explicitly legitimizing these assumptions.

Chapter 12

Geographic Routing and Mobility

Never confuse movement with action.
Ernest Hemingway (1899–1961)

The dynamic characteristics of ad hoc networks are based on two important factors: On the one hand, wireless links are inherently less stable than wired ones; on the other hand, network nodes are in many scenarios potentially mobile. The analysis of the geographic routing algorithms in the previous chapters assumes that routing occurs much faster than network dynamics. Technically, the consequence of this assumption is that the network graph remains static throughout the execution of the considered routing algorithm. While it appears reasonable to assume so for networks consisting of relatively few nodes, mobility and edge dynamics can become a non-negligible issue in large-scale ad hoc networks.

As in the previous chapters, differentiation between average-case and worst-case considerations appears to be reasonable also in the context of mobility and dynamics of ad hoc networks. If it is almost always difficult to formulate what should be understood by "average-case" behavior of a system, this inherent difficulty is particularly obvious as regards edge dynamics or node mobility patterns in ad hoc networks. What is a typical link quality characteristic of a wireless connection? What does typical motion of a network node look like? These are questions that need to be answered for the study of average-case networks. While some answers can be given to the first of these questions based on physical models of signal propagation, finding an answer to the second question appears to be more difficult, particularly as no large-scale ad hoc networks are in operation yet. A common approach to this problem is the definition of a node movement pattern, such as the random waypoint model—where nodes choose a random destination and move there on a straight line with an also randomly

chosen velocity, repeating this procedure as soon as reaching the destination—, a Brownian motion model—where nodes repeatedly take small steps in random directions—, or a random walk on a grid or a similar discretization of the Euclidean plane. A different approach could consist in focusing on scenarios where real-world motion patterns are available, although not (yet) in the context of ad hoc networks. A typical example for such an approach would be the car fleet ad hoc network scenario, where car motion patterns are present from traffic analysis and where the cars could later be "equipped" with radio devices, for instance for the purpose of simulation.

Greedy Routing as described in Chapter 4 not only promises to be an efficient routing technique in dense networks, but particularly in ad hoc networks with average-case node mobility. Greedy Routing can to a certain extent be compared with source routing, the source addressing the message with the destination position. Compared to traditional source routing—where the source specifies the complete route to be taken by the message and where absence of a single node therefore breaks the route—, Greedy Routing can be expected to be significantly more resilient to moving intermediate nodes: As long as the destination remains relatively static, it will be reached (almost) regardless of mobility of nodes located between the source and the destination. In this sense, also the GOAFR and GOAFR$^+$ algorithms discussed in Chapter 7 can be made more resilient to average-case node mobility—while worst-case node mobility is an issue of its own and will be discussed later. These algorithms, as they are described in Chapter 7, rely on a visited face boundary to remain topologically static throughout its traversal. In particular this means that—in a degenerate case—a single node, having transferred the message along a face boundary when moving slightly and consequently leaving the considered face boundary, can prevent any of the above algorithms from operating correctly. In many cases the algorithm will succeed in continuing towards the destination if the notion of "visiting a node for a second time" is replaced by "visiting a region around a node for a second time", albeit not always being able to preserve its performance guarantees. In a similar spirit, restricted flooding (cf. Chapter 10) can be combined with GOAFR($^+$), for instance to find a destination having recently changed its location. The necessity and effectiveness of such measures can however not be assessed in a general sense and highly depend on the considered node mobility pattern.

The aim when considering worst-case graph dynamics—in contrast to average-case dynamics—would be to guarantee that a routing algorithm always reaches the destination, in a first attempt even regardless of its cost. Closer consideration however shows that worst-case node mobility can bring about a host of virtually unsolvable problems. Not only "unreasonably degenerate" cases such as arbitrarily fast nodes "abducting" the message from the remainder of the network, but also less artificial examples, such as island forming processes as illustrated in Figure 12.1, have to be excluded in order to make worst-case mobility accessible to analysis. Furthermore, models traditionally employed for the analysis of dynamic graph structures [32] do not lend themselves to the pur-

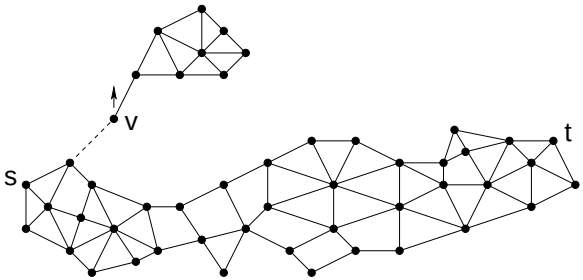

Figure 12.1: If the bridge node v moves north after being traversed by the routed message (also in northern direction), the dashed edge may disappear due to its increasing length, and consequently the message will be isolated on the newly formed network island containing v.

pose of modeling dynamic ad hoc networks, as these models study insertion and deletion of single nodes or edges and thus only insufficiently reflect the characteristics of ad hoc networks. It appears that the study of ad hoc networks with worst-case node mobility forms a classic instance of the dilemma of finding a model which on the one hand reflects reality and on the other hand admits of an analysis beyond trivialities.

At first sight, a model where network nodes are static and exclusively edges can dynamically appear or disappear during the execution of an algorithm seems to be a good first step towards a dynamic ad hoc network model.[1] A more careful consideration of this model however leads to the conclusion that similar effects as the above island forming can occur. To allow of reasonable analysis, graph dynamics, be it edge dynamics or node mobility, have to be described in a restrictive manner. For instance, it could be demanded that edge dynamics are such that a subset of all edges remain in the graph for a certain period of time during algorithm execution. This however does not appear to solve the problem. Requiring for example that at all times the current location of the message is connected to the part of the network containing the destination does not avoid island forming, as the message can be isolated with alternatingly appearing and disappearing "bridge" edges connecting the (virtual) island to the remainder of the graph. Strengthening this requirement, demanding a spanning tree of the network to remain in the graph throughout algorithm execution, on the other hand helps to ensure that the message reaches the destination, appears however to be an unrealistic assumption.

[1] Interestingly, this corresponds to a possible model for an urban area automobile ad hoc network scenario on a higher level of abstraction, where network nodes reflect street intersections and edges stand for streets connecting the intersections for communication depending on the current traffic, that is the presence of communication relay nodes.

In summary, modeling of mobility in ad hoc networks appears to currently be an issue with many yet unanswered questions. While scenario-driven simulation based on real-world mobility patterns is probably the most reasonable approach for the study of average-case mobility in ad hoc networks, it appears to be unclear what a model should look like which reflects the characteristics of a mobile ad hoc network and simultaneously admits of algorithm analysis with worst-case mobility.

Part II

Topology Control

Chapter 13

Topology Control: An Introduction

The national budget must be balanced. The public debt must be reduced; the arrogance of the authorities must be moderated and controlled. Payments to foreign governments must be reduced if the nation doesn't want to go bankrupt. People must again learn to work instead of living on public assistance.
Cicero (106 BC – 43 BC)

In a very general sense, topology control can be considered the task of—given a communication network graph—constructing a subgraph with certain desired properties. This type of topology control was employed in several occasions in the first part of the dissertation, for instance constructing a planar subgraph for geographic routing, generating a routing backbone with bounded degree, and controlling the number of nodes or edges in a given area. The XTC algorithm presented in Chapter 14 adopts this notion of topology control, analyzing the question of how a subgraph structure with provable properties, such as graph connectivity, low node degree, planarity, or the spanner property, can be constructed by means of a light-weight algorithm, particularly attempting to do without worrisome unrealistic assumptions.

The subsequent chapters will move their focus to a more specific understanding of topology control. In particular, energy being among the most critical resources in ad hoc networks, topology control will be considered a method to reduce energy consumption. The basic mechanism employed to this end consists in having the network nodes reduce their transmission power levels in a coordinated and controlled manner. On the one hand, nodes can thus save energy directly, by transmitting messages with lowered power and reducing the energy thereby consumed. Also from a more collaborative perspective, the total energy consumed when sending a message over several short edges is lower than when transmitting it directly from the source to the destination, as the energy

required to send a message over a wireless link grows at least quadratically with the length of the link. More importantly, however, reducing transmission power and thus the extension of transmission ranges can contribute to reduction of interference in the network. Topology control in this sense also indirectly lowers energy consumption by reducing interference, the probability of message collision, and consequently the number of retransmissions. At the same time, the transmission ranges may not be made too small, as the network could lose connectivity and prevent proper communication. Topology control can therefore be considered a trade-off between energy conservation and the preservation of vital network properties.

A third perspective is reflected in that sometimes also the construction of node clusters and dominating sets of nodes is considered topology control. Apart from Chapter 14, however, this second part of the dissertation will be restricted to the study of topology control based on transmission power reduction.

If interference reduction has often been mentioned as one of the main goals of topology control, previous work (with few exceptions) has generally stated interference to be lowered implicitly, in particular as a consequence of low node degree of the constructed topology. In this dissertation we will attempt to analyze this statement in depth. Specifically, a first step towards this end consists in defining clearly what is understood by interference. Chapter 15 comprises such a definition of an explicit interference model and a discussion showing that almost all previously proposed topology control algorithms—even if constructing topologies with bounded degree—do not always effectively reduce interference.

The interference model defined in Chapter 15 is based on the number of nodes affected by communication over a given link. In other words, this model focuses on the *sending* process of message transmission. It can be argued that such a perspective puts the cart before the horse, as interference and in particular message collisions actually occur at the intended *receiver* of a message. Such a receiver-centric interference model will be discussed in Chapter 16 in the context of data gathering in sensor networks and will be extended for application in general ad hoc networks in Chapter 17.

A different approach to the issue of interference reduction will be taken in Chapter 18. Besides defining a model of interference in cellular networks, this chapter will formalize the task of lowering interference as a combinatorial optimization problem tackled by linear programming. Although the context of cellular networks go beyond the scope of this dissertation in a strict sense, we believe that this approach deserves being discussed, as it augments the previous chapters by a new perspective on interference reduction.

Interference models forming a key point of this second part of the dissertation, Chapter 19 will try to admit a "look behind the scenes", showing that a plethora of interference models can be defined, the most reasonably justifiable ones of which having been analyzed in the preceding chapters.

Finally, it is to be emphasized that this part of the dissertation will present various *approaches* to the task of reducing interference in ad hoc and sensor

networks, particularly focusing on interference models in graph representations of networks. Many questions remain yet unanswered. In particular, the consequences and effects of our theoretical models and analytical studies with respect to practical networks is predictable with difficulty only. We consider our work to be a first step towards understanding the complex interplay between interference and energy efficiency in sensor networks.

Related Work

The assumption that nodes are distributed randomly in the plane according to a uniform probability distribution formed the basis of pioneering work in the field of topology control in ad hoc networks [47, 101].

Later proposals adopted constructions originally studied in computational geometry, such as the Delaunay Triangulation [48], the minimum spanning tree [94], the Relative Neighborhood Graph [11], or the Gabriel Graph [97]. Most of these contributions mainly considered energy-efficiency of paths preserved by the resulting topology, whereas others exploited the planarity property of the proposed constructions for geographic routing.

The Delaunay Triangulation and the minimum spanning tree not being computable locally and thus not being practicable, a next generation of topology control algorithms emphasized locality. The CBTC algorithm [109] was the first construction to simultaneously focus on several desired properties, in particular being an energy spanner with bounded degree. This process of developing local algorithms featuring more and more properties was continued, partly based on CBTC, partly based on local versions of classic geometric constructions such as the Delaunay Triangulation [71] or the minimum spanning tree [70]. One of the most recent such results is a locally computable planar constant-stretch distance (and energy) spanner with constant-bounded node degree [108]. Another thread of research takes up the average-graph perspective of early work in the field; [10] for instance shows that the simple algorithm choosing the k nearest neighbors works surprisingly well on such graphs.

Yet another aspect of topology control is considered by algorithms trying to form clusters of nodes. Most of these proposals are based on (connected) dominating sets (cf. Related Work in Chapter 2) and focus on locality and provable properties. Cluster-based constructions are commonly regarded a variant of topology control in the sense that energy-consuming tasks can be shared among the members of a cluster.

Topology control having so far mainly been of interest to theoreticians, first promising steps are being made towards exploiting the benefit of such techniques also in practical networks [58].

Where the XTC algorithm [110] discussed in Chapter 14 focuses on topology construction based on minimal assumptions about the capabilities of nodes and signal propagation characteristics, the subsequent chapters turn their attention to interference. As mentioned earlier, reducing interference—and its

energy-saving effects on the medium access layer—is one of the main goals of topology control besides direct energy conservation as a consequence of transmission power restriction. Astonishingly however, all the above topology control algorithms at the most implicitly try to reduce interference. Where interference is mentioned as an issue at all, it is maintained to be confined at a low level as a consequence of sparseness or low degree of the resulting topology graph.

A notable exception to this is [74] defining an explicit notion of interference. Based on this interference model between edges, a time-step routing model and a concept of congestion is introduced. It is shown that there are inevitable tradeoffs between congestion, power consumption and dilation (or hop-distance). For some node sets, congestion and energy are even shown to be incompatible.

The interference model proposed in [74] is based on current network traffic. The amount and nature of network traffic however highly depends on the chosen application. Since usually no a priori information about the traffic in a network is available, a static model of interference depending solely on node constellations is consequently desirable. Such a traffic-independent notion of interference was introduced in [16] and will be presented in Chapter 15. As we will show, the above statement that graph sparseness or small degree implies low interference is misleading. The interference model described in Chapter 15—further analyzed in [75]—builds on the question of how many nodes are affected by communication over a given link. As will also be discussed in more depth, this sender-centric perspective can however be accused to be somewhat artificial and to poorly represent reality, interference occurring at the intended *receiver* of a message. Furthermore, this interference measure will be shown to be susceptible to drastic effects even if single nodes are added to or removed from a network.

An attempt to correct for this deficiency was made in [37], which will be presented and discussed in Chapter 16. As we will show, this work defines a receiver-centric concept of interference in the context of data-gathering structures in sensor networks. The issue of energy efficiency in sensor networks [2, 20, 27]—particularly extending network lifetime—has been mainly studied with respect to optimal sensor placement and energy-efficient routing. Recently also the fact that certain types of sensed data allow for aggregation at sensor nodes [44] and the existence of redundancy in acquired information [21, 105]—for instance correlation between sensed data depending on the distance between sensors—has been considered.

The approach originally presented in [104] and discussed in Chapter 17 goes beyond sensor networks by defining and employing a suitable robust interference model for the analysis of topology control in ad hoc networks in general.

Chapter 18 will address interference in cellular networks by formulating and attempting to solve a combinatorial optimization problem [63]. Interference issues in cellular networks have been studied since the early 1980s in the context of frequency division multiplexing: The available network frequency spectrum is divided into narrow channels assigned to cells in a way to avoid interference conflicts. In particular, two types of conflicts can occur, adjacent cells using the same channel (cochannel interference) and insufficient frequency distance

between channels used within the same cell (adjacent channel interference). Maximizing the reuse of channels avoiding these conflicts was generally studied by means of the combinatorial problem of conflict graph coloring using a minimum number of colors. The settings in which this problem was considered are numerous and include hexagon graphs, geometric intersection graphs (such as unit disk graphs), and planar graphs, but also (non-geometric) general graphs. In addition, both static and dynamic (or on-line) approaches were studied [80]. The fact that channel separation constraints can depend on the distance of cells in the conflict graph was studied by means of graph labeling [51]. The problem of frequency assignment is tackled in a different way in [33] exploiting the observation that in every region of an area covered by the communication network it is sufficient that exactly one base station with a unique channel can be heard. As mentioned, all these studied models try to avoid interference conflicts occurring when using frequency division multiplexing. In contrast, the problem described in Chapter 18 assumes a different approach in aiming at interference reduction by having the base stations choose suitable transmission power levels.

The problem of reducing interference is formalized in Chapter 18 in a combinatorial optimization problem named *Minimum Membership Set Cover*. As suggested by its name, at first sight its formulation resembles closely the long-known and well-studied *Minimum Set Cover (MSC)* problem, where the number of sets chosen to cover a collection of given elements is to be minimized [54]. That the MMSC and the MSC problems are however of different nature can be concluded from the following observation: For any MSC instance consisting of n elements, a greedy algorithm approximates the optimal solution with an approximation ratio at most $H(n) \leq \ln n + 1$ [54], which has later been shown to be tight up to lower order terms unless $NP \subset TIME(n^{O(\log \log n)})$ [34, 72] (cf. Chapter 18 for details). For the MMSC problem in contrast, there exist instances where the same greedy algorithm fails to achieve *any* nontrivial approximation of the optimal solution.

Chapter 14

Lightweight Topology Control

He who would travel happily must travel light.
Antoine de Saint-Exupéry (1900–1944)

The currently best proposals of topology control structures feature an impressive collection of properties. Maybe the foremost drawback of these algorithms is however that they require many unrealistic assumptions: First, most algorithms assume that all the nodes know their exact positions. Second, the algorithms assume that the world is flat and without obstacles to the propagation of radio signals. In this chapter we will present the XTC[1] topology control algorithm that works i) without position knowledge and ii) even in a mountainous and obstructed environment. Surprisingly, the XTC algorithm features many of the relevant properties of topology control while being simpler than most previous proposals. Not being based on unrealistic assumptions, XTC is probably among the topology control algorithms lending themselves most for practical implementation.

For two communicating ad hoc network nodes u and v, the energy consumption of their communication grows at least quadratically with their distance. Having one or more relay nodes between u and v therefore helps to save energy. Among the primary targets of a topology control algorithm is to abandon long-distance communication links and instead route a message over several small (energy-efficient) hops. For this purpose, each node in the ad hoc network chooses a "handful" of "close-by" neighbors "in all points of the compass" (the details being explained later). Clearly, nodes cannot abandon links

[1]To date, the inventors of the algorithm have not yet been able to agree on the meaning of the letter "X" in "XTC". The candidate list comprises terms such as "exotic", "extreme", "exceptional", or "exemplary", but also "extravagant" or even "extraterrestrial". Consensus has however been achieved concerning the pronunciation of the algorithm name.

to "too many" far-away neighbors in order to prevent the ad hoc network from being partitioned or the routing paths from becoming non-competitively long. In general, there is a trade-off between network connectivity and sparseness.

Let the graph $G = (V, E)$ denote the ad hoc network before running the topology control algorithm, with V being the set of ad-hoc nodes, and E representing the set of communication links. There is a link (u, v) in E if and only if the two nodes u and v can communicate directly. Running the topology control algorithm will yield a sparse subgraph $G_{tc} = (V, E_{tc})$ of G, where E_{tc} is the set of remaining links. The resulting topology G_{tc} should have the following properties:

Property 1 (Symmetry) The resulting topology G_{tc} should be symmetric, that is, node u is a neighbor of node v if and only if node v is a neighbor of node u.

Asymmetric communication graphs are impractical because many communication primitives become unacceptably complicated. A simple acknowledgement message confirming the receipt of a message, for example, is already a nightmare in an asymmetric graph [92].

Property 2 (Connectivity) Two nodes u and v are connected if there is a path from u to v, potentially through multiple hops. If two nodes are connected in G, then they should still be connected in G_{tc}.

Although a minimum spanning tree (MST) is a sparse connected subgraph, it is often not considered a good topology since close-by nodes in the original graph G might end up being far away in G_{tc} (G being a ring, for instance). Therefore Property 2 is usually strengthened:

Property 2+ (Spanner) For any two nodes u and v, if the optimal path between u and v in G has cost c, then the optimal path between u and v in G_{tc} has cost $f(c)$. If $f(c)$ is bounded from above by a linear function in c, the graph G_{tc} is called a spanner with linearly bounded stretch.

As described in Chapter 3, different cost metrics can be studied in the context of ad hoc networks. In this chapter the Euclidean distance and the energy metric will be considered as introduced in Definition 3.2. Remember that for both metrics the cost of a path is defined to be the summed up cost of all links in the path.

As mentioned, the primary target of a topology control algorithm is to abandon long-distance neighbors, or more formally:

Property 3 (Sparseness) The remaining graph G_{tc} should be sparse, that is, the number of links should be in the order of the number of nodes: $|E_{tc}| \in O(|V|)$.

This reflects that not too many close-by nodes must be chosen, which can be expected to reduce interference and thus to save energy in average-case net-

works. Since there still might be some nodes with many neighbors (for instance a star graph), also Property 3 features an improved version.

Property 3+ (Low Degree) Each node in the remaining graph G_{tc} has a small number of neighbors. In particular, the maximum degree in the graph G_{tc} should be bounded from above by a constant.

Since connectivity and sparseness run against each other, topology control has been a thriving research area. In addition to the properties 1, 2, and 3, often secondary targets can be found. For instance it is popular (and often for free) to require the resulting graph to be planar in order to allow for geographic routing as discussed in the first part of this dissertation.

This chapter features three major contributions. First, it is agreed upon that the subgraph G_{tc} should not be computed by a heavyweight *global* algorithm, but instead with a *distributed* algorithm. It is often argued that an algorithm—in order to be able to cope with mobility—should not only be *distributed* but even *local*: Each node is allowed to exchange messages with its neighbors a few times whereafter it must decide which links it wants to keep. Many naive topologies, such as the minimum spanning tree, can provably not be computed locally and are therefore not realistic. To the best of our knowledge, we present the fastest algorithm so far, where each node only communicates with its neighbors twice.

A second often made assumption is that the ad hoc network nodes are represented by points in a Euclidean plane. Two nodes are connected in the original graph G if and only if their Euclidean distance is at most 1 (a normalized transmission radius), resulting in a unit disk graph (cf. Definition 3.1). As good a first step towards understanding ad hoc network algorithms the employment of unit disk graphs may be, this model is not practical, as it is based on several assumptions:

i) All nodes are homogeneous.

ii) Antennas are perfect isotropic radiators, such that all transmission radii are equal.

iii) Attenuation is uniform, that is, the Euclidean plane is flat and free of blocking objects, such as walls. Radio propagation is as in vacuum.

Especially Assumption iii) is questionable in any realistic environment. We believe that ad hoc network algorithms should work, or in other words be correct, also in a more realistic environment that goes beyond the unit disk graph. While Chapter 10 displays an attempt to generalize the unit disk graph, this chapter even allows the original communication graph G not to comply with any of the above assumptions. Instead, G is a general graph without any geometric assumptions. For instance—as opposed to many other algorithms—XTC also works correctly if nodes are located in three-dimensional space, as in a building. When studying efficiency (not correctness), analytically and by simulation, we

will make geometric assumptions to prove stronger results that compare better to related work. To the best of our knowledge we present the first topology control algorithm that works for general graphs.

A third assumption that is commonly made is that the nodes have detailed information about their neighbors. It is often assumed that all the nodes know their exact coordinates in the plane, for instance by means of a global positioning system (GPS). The most notable exception is the cone-based topology control (CBTC) algorithm [109], where nodes conclude information about their neighbors merely based on their relative signal strength and the signal arrival angle. It is in this third respect that this chapter's main contribution lies. For the correctness of the algorithm it is sufficient that the network nodes order their neighbors according to a general concept of link quality.

In a sense, the results presented in this chapter can be considered a paradigm shift in topology control. Where recent research tried to improve existing algorithms by enhancing them with various new features (and thus rendering the algorithms more complicated), we will present an algorithm that is simpler, faster, and works without unrealistic assumptions.

This chapter is organized as follows: After providing preliminary definitions in Section 14.1, we will describe the XTC algorithm in Section 14.2. For illustration, Section 14.3 will prove XTC's properties when employed on Euclidean and unit disk graphs. The behavior of the algorithm on general weighted graphs is the subject of Section 14.4. The subsequent section will provide an evaluation of XTC on average-case random graphs.

14.1 Preliminaries

This section will provide formal definitions of basic concepts essential for the understanding of this chapter.

In a *weighted graph* $G = (V, E)$, every edge $(u, v) \in E$ is attributed a weight ω_{uv}. When referring to a weighted graph, we assume throughout this chapter that the weights are symmetric: $\omega_{uv} = \omega_{vu}$.

The nodes of a *Euclidean graph* are assumed to be located in a Euclidean plane. Furthermore the edge weight of an edge (u, v) is defined to be $\omega_{uv} = |uv|$, where $|uv|$ is the Euclidean distance between the nodes u and v. Note that the definition of Euclidean graphs does not contain a statement on the existence of certain edges.

Strongly related to edge weights is the *cost of an edge*. The cost of an edge $c(u, v)$ can be considered to represent the effort an algorithm is required to expend in order to send a message over (u, v). The cost metrics we will study in this chapter correspond to the cost definition in Chapter 3, including the most popular link, Euclidean, and energy metrics as well as the definition of the cost of a *path*.

14.2 XTC Algorithm

In this section we will describe our topology control algorithm XTC. The algorithm consists of three main steps:

I) Neighbor ordering,

II) neighbor order exchange, and

III) edge selection.

In the first step, each network node u computes a total order $\prec_u$ over all its neighbors in the network graph G. From an abstract point of view, this order is intended to reflect the quality of the links to the neighbors. A node u will consider its neighbors in G (in the third step of the algorithm) according to $\prec_u$ ordered with respect to decreasing link quality: The link to a neighbor appearing early in the order $\prec_u$ is regarded as being of higher quality than the link to a neighbor placed later in $\prec_u$. A neighbor w appearing before v in order $\prec_u$ is denoted as $w \prec_u v$. For illustration we will assume in Section 14.3 that $\prec_u$ corresponds to the order of the neighbors' Euclidean distances from u. It is however conceivable that the neighbor order reflects a much more general notion of link quality, such as signal attenuation or packet arrival rate.

In the second step, the neighbor order information is exchanged among all neighbors. Typically, a node u broadcasts its own neighbor order while receiving the orders established by all of its neighbors.

During the third step, which does not require any further communication, each node locally selects those neighboring nodes which will form its neighborhood in the resulting topology control graph, based on the previously exchanged neighbor order information. For this purpose, a node u traverses $\prec_u$ with decreasing link quality: "Good" neighbors are considered first, "worse" ones later. Informally speaking, a node u only builds a direct communication link to a neighbor v if u has no "better" neighbor w that can be reached more easily from v than u itself.

Although the XTC algorithm is executed at all nodes, the detailed description as shown in the above box assumes the point of view of a node u. Lines 1 and 2 correspond to Steps I) and II). Lines 3-11 define Step III) in more detail: First the two neighbor sets N_u and $\widetilde{N}_u$ are initialized to be empty. Now the neighbor ordering $\prec_u$ established in Line 1 is traversed in increasing order. In Line 7, the neighbor order $\prec_v$ of the currently considered neighbor v is examined: If any of u's neighbors w already processed appears in v's order before u ($w \prec_v u$), node v is included in $\widetilde{N}_u$ (Line 8); otherwise v is added to N_u (Line 10).

After completion of the algorithm, the set N_u contains u's neighbors in the topology control graph G_{XTC}. More formally, the edge set E_{XTC} of the graph $G_{XTC} = (V, E_{XTC})$ is $E_{XTC} = \{(u,v) | \exists u\colon v \in N_u\}$.

In the algorithm as described above, each node constructs in Step I) a total order over all its neighbors in G. In a variant of the algorithm a node u could

XTC Algorithm

1: Establish order $\prec_u$ over u's neighbors in G

2: Broadcast $\prec_u$ to each neighbor in G; receive orders from all neighbors

3: Select topology control neighbors:
4: $N_u := \{\}$; $\widetilde{N}_u := \{\}$
5: while ($\prec_u$ contains unprocessed neighbors) {
6: $v :=$ least unprocessed neighbor in $\prec_u$
7: if $(\exists\, w \in N_u \cup \widetilde{N}_u : w \prec_v u)$
8: $\widetilde{N}_u := \widetilde{N}_u \cup \{v\}$
9: else
10: $N_u := N_u \cup \{v\}$
11: }

apply a growing radius technique—starting with the "best" neighbor—to decide on a neighbor v's inclusion in N_u or $\widetilde{N}_u$—based on $\prec_v$—immediately when identifying v as the next "worse" neighbor found so far. Applying such interleaving of steps I), II) and III), u could terminate earlier, that is, as soon as having found "enough" neighbors (where Theorem 14.3 would provide a termination criterion in the case of G being a unit disk graph).

Property 1 as described in the introduction of this chapter, that is symmetry of the resulting graph, often has to be enforced by topology control algorithms (for instance by a propose-accept cycle) [106, 109]. The following theorem shows that, in contrast, XTC is guaranteed to "automatically" compute a graph with Property 1 without any assumptions whatsoever on the neighbor orders:

Theorem 14.1 (Symmetry). *The edges in G_{XTC} are symmetric: A node u includes a neighbor v in N_u if and only if v includes u in N_v.*

Proof. Assume for the sake of contradiction that u includes v in N_u (Assumption 1), whereas v does not include u in N_v (Assumption 2). According to Assumption 2, there exists a node $w \in N_v \cup \widetilde{N}_v$ when v decides to include u in $\widetilde{N}_v$ in Line 8 ($w \prec_v u$), such that $w \prec_u v$ (Line 7). Since $w \prec_u v$ holds, $w \in N_u \cup \widetilde{N}_u$ at the point of time when u decides about v's inclusion in N_u; together with $w \prec_v u$, it follows that v is included in $\widetilde{N}_u$, which is a contradiction to Assumption 1. □

14.3 XTC on Euclidean Graphs

The main purpose of this section is to provide an illustration of the graph resulting from the topology control algorithm. We make three assumptions:

i) Every node u has a unique identifier id_u. The identifiers are comparable, that is, there exists a total order "$<$" defined over the set of all identifiers.

ii) The nodes are placed in a Euclidean plane.

iii) Every edge (u, v) is attributed a weight defined to be the triple

$$(|uv|, \min(id_u, id_v), \max(id_u, id_v)),$$

where $|uv|$ is the Euclidean distance between nodes u and v. The neighbor orders computed in Step I) of the XTC algorithm are based on the lexicographic order[2] of these edge weights, that is

$$w \prec_u v \iff (|uw|, \min(id_u, id_w), \max(id_u, id_w)) < (|uv|, \min(id_u, id_v), \max(id_u, id_v))$$

Assumption i) is common in the context of distributed algorithms and viable for practical networks. Assumption ii)—although often made in order to model ad hoc networks—is less realistic; nevertheless we adopt this assumption in this section for the sake of illustration.

Assumption iii) can for instance be realized by having each node initially transmit a control signal together with a message containing information on the control signal transmission power. With the additional assumption that the employed antennas are isotropic and that the signal can propagate without obstruction, the control signal receivers can compute an order over the Euclidean distances to the senders from the receive and transmission power levels. If all nodes send with equal transmission power, the order $\prec_u$ is even equivalent to the relative order of the received signal strengths sensed at a node u.

The following theorem proves Property 2 as defined in the introduction of this chapter, that is connectivity of the topology control graph G_{XTC}. Note that this theorem does not require G to be a unit disk graph; G being a Euclidean Graph is sufficient.

Theorem 14.2 (Connectivity). *Given a Euclidean Graph G, two nodes u and v are connected in G_{XTC} if and only if they are connected in G. Consequently, the graph G_{XTC} is connected if and only if G is connected.*

Proof. Since XTC exclusively considers edges in G, u and v can only be connected if they are connected in G. In order to prove the opposite direction of the above equivalence, we assume for contradiction that G_{XTC} contains at least one pair of non-connected nodes that are connected in G. Consider the pair u, v with minimum value $(c_d(p^*(u, v)), \min(id_u, id_v), \max(id_u, id_v))$—where $c_d(p^*(u, v))$ is the Euclidean cost of the shortest path connecting u and v on G—among all pairs of nodes u and v that are not connected in G_{XTC} but connected in G. The nodes u and v must be connected directly by the edge (u, v) in G; otherwise a different pair of nodes w, x lying on the path connecting u and v would have a value

[2]The lexicographic order of two triples is defined according to the order of the first components, or—if the first components are equal—according to the second components, or—if both the first and the second components, respectively, are equal—according to the third components. Formally: $(a_1, b_1, c_1) < (a_2, b_2, c_2) \iff (a_1 < a_2) \vee ((a_1 = a_2) \wedge (b_1 < b_2)) \vee ((a_1 = a_2) \wedge (b_1 = b_2) \wedge (c_1 < c_2))$.

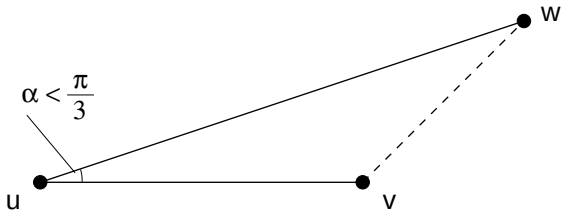

Figure 14.1: As described in the proof of Theorem 14.3, if $\alpha < \pi/3$ and $|uv| \leq |uw|$, it follows that $|vw| < |uw|$.

$c_d(p^*(w,x))$ less than $c_d(p^*(u,v))$, and u, v would not be the pair with minimum $c_d(p^*(u,v))$. Since the edge (u,v) is in G, the cost of the shortest path connecting u and v is their Euclidean distance: $c_d(p^*(u,v)) = |uv|$. According to the assumption, u includes v in $\widetilde{N}_u$, that is, at the moment u decides so, there is a node $w \in N_u \cup \widetilde{N}_u$ such that $w \prec_v u$. Since $w \in N_u \cup \widetilde{N}_u$, we also have $w \prec_u v$, or $(|uw|, \min(id_u, id_w), \max(id_u, id_w)) < (|uv|, \min(id_u, id_v), \max(id_u, id_v))$. Since $(|uv|, \min(id_u, id_v), \max(id_u, id_v))$ is the least such value for any pair of non-connected nodes in G_{XTC}, and as u and w are connected in G (w is contained in $\prec_u$), u and w must also be connected in G_{XTC}. For the same reason and since $w \prec_v u$, also v and w must be connected in G_{XTC}, which contradicts the assumption that u and v are not connected in G_{XTC}. □

For the remainder of this section we will now assume that G is a unit disk graph. The following theorem proves that in this case G_{XTC} does not only feature sparseness (Property 3 as described in the introduction of this chapter), but even bounded degree (Property 3+):

Theorem 14.3 (Bounded Degree). *Given a unit disk graph G, G_{XTC} has node degree at most 6.*

Proof. We prove that no two adjacent edges in G_{XTC} enclose an angle less than $\pi/3$, from which the theorem follows. Assume for contradiction that the two edges (u,v) and (u,w) enclose an angle $\alpha < \pi/3$ at node u. Furthermore let v be u's neighbor that was included in N_u before w, that is $v \prec_u w$ or $(|uv|, \min(id_u, id_v), \max(id_u, id_v)) < (|uw|, \min(id_u, id_w), \max(id_u, id_w))$. Since $|uv| \leq |uw|$ and $\alpha < \pi/3$, it follows that $|vw| < |uw|$ (cf. Figure 14.1). G being a unit disk graph, also the edge (v,w) is in G, as $|vw| < |uw| \leq 1$. Consequently $v \prec_w u$, implying that u included w in $\widetilde{N}_u$, which is however a contradiction to the assumption that the edge (u,w) is in G_{XTC}. □

As an additional property, G_{XTC} contains no two intersecting edges, which allows its employment for geographic routing as discussed in the first part of this dissertation.

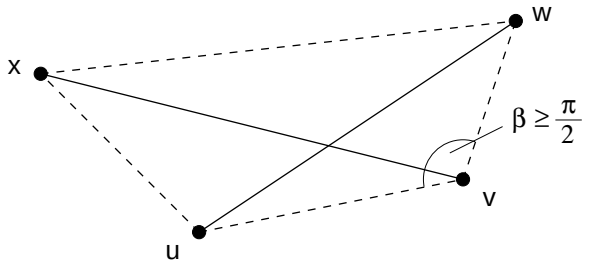

Figure 14.2: The quadrangle $uvwx$ in the proof of Theorem 14.4 contains at least one angle $\beta \geq \pi/2$. It follows that if G is a unit disk graph, the edge (u, w) is not contained in G_{XTC}.

Theorem 14.4 (Planarity). *Given a unit disk graph G, G_{XTC} is planar, that is, it contains no two intersecting edges.*

Proof. Suppose for the sake of contradiction that the two edges (u, w) and (v, x) intersect in G_{XTC} (cf. Figure 14.2). Of the quadrangle $uvwx$ at least one angle has size not less than $\pi/2$. Let this angle be without loss of generality β adjacent to node v. Since $\beta \geq \pi/2$, $|uv| < |uw|$ and $|vw| < |uw|$. When node u considered the inclusion of w in N_u, v was consequently already in $N_u \cup \widetilde{N}_u$ and $v \prec_w u$, causing u to include w in $\widetilde{N}_u$, which is however a contradiction to the assumption that the edge (u, w) is in G_{XTC}. □

In the following theorem we will describe the relationship between G_{XTC} and the Relative Neighborhood Graph of G [102]. The Relative Neighborhood Graph is defined to contain all edges $(u, v) \in G$, such that there exists no node w with $|uw| < |uv| \wedge |vw| < |uv|$ (cf. Figure 14.4(a)).

Theorem 14.5. *Given a unit disk graph G, G_{XTC} is a subgraph of the Relative Neighborhood Graph computed on G. If G contains no node having two or more neighbors at exactly the same distance, G_{XTC} is identical to the Relative Neighborhood Graph.*

Proof. The subgraph relationship follows from the fact that if G_{XTC} contains an edge (u, v), there exists no node w with $|uw| < |uv| \wedge |vw| < |uv|$, which implies that (u, v) is also contained in the Relative Neighborhood Graph RNG. Furthermore XTC excludes an edge (u, v) that is preserved in RNG—there is no w with $|uw| < |uv| \wedge |vw| < |uv|$—only if there exists a node w with $w \prec_u v \wedge w \prec_v u$, which is in total possible only if $|uw| = |uv|$ and the enclosing angle $\angle vuw \leq \pi/3$ or $|vw| = |uv|$ and the enclosing angle $\angle uvw \leq \pi/3$. □

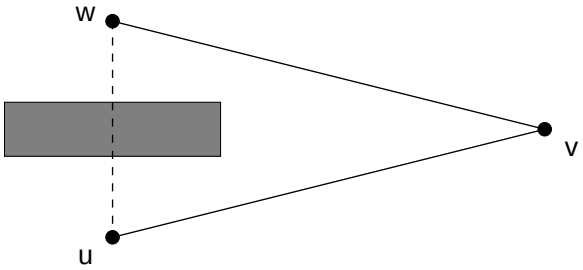

Figure 14.3: The edge weights ω_{uv}, ω_{uw}, and ω_{vw} reflect that signal propagation between u and w is impaired by a physical obstacle (wall, building, hill): $\omega_{uv} < \omega_{uw}$ and $\omega_{vw} < \omega_{uw}$. In contrast to typical topology control algorithms based on node positions, XTC does not include the edge (u, w) in its resulting graph, but exploits that a better connection exists via node v.

14.4 XTC on General Weighted Graphs

In realistic ad hoc networks, nodes are not located in a plane and received signal strength does not only depend on the distance to the sender, but above all on physical obstacles between sender and receiver. As one of the main properties of such real ad hoc networks, however, symmetry of physics is preserved: The attenuation factor of a link between two network nodes is identical to signal propagation in either direction. Accordingly, an ad hoc network can be modeled by a weighted graph, where each edge is attributed a (symmetric) weight (cf. Section 14.1) representing the corresponding signal attenuation factor (see Figure 14.3). More abstractly, the edge weights can be considered qualities of links between node pairs. Assuming isotropic antennas, a node can obtain its neighbor order with a technique similar to the one described in Section 14.3 by sending a control signal. If the edge weights are considered link quality indicators in a more general sense, these weights and consequently the neighbor ordering can be established by exchange of probe messages.

In this section we will show that the XTC algorithm computes a connected topology even in general weighted graphs (with symmetric edge weights) modeling realistic ad hoc networks. We assume that the neighbor order of a node u—as employed in the algorithm—corresponds to the order over the weights of the edges adjacent to u.

Theorem 14.6 (Connectivity). *Given a general weighted graph G, two nodes u and v are connected in G_{XTC} if and only if they are connected in G. Consequently, the graph G_{XTC} is connected if and only if G is connected.*

Proof. This theorem can be proved in analogy to the proof of Theorem 14.2, substituting edge weights for Euclidean distances. □

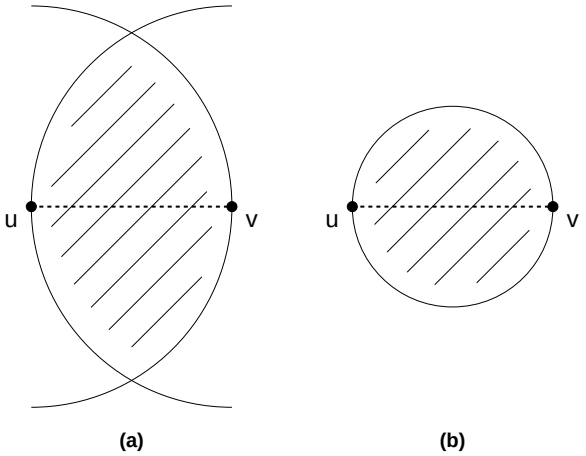

Figure 14.4: Definitions of the Relative Neighborhood Graph (a) and the Gabriel Graph (b). In the Relative Neighborhood Graph, the edge (u, v) exists if and only if the shaded lune (excluding its boundary) does not contain a third node. In the Gabriel Graph, the edge (u, v) exists if and only if the shaded circle (including its boundary) does not contain a third node (cf. Figure 3.1).

What furthermore can be stated with respect to G_{XTC}'s sparseness (Property 3)—if G is a general weighted graph—is that G_{XTC} cannot contain cycles of length three:

Theorem 14.7. *Given a general weighted graph G, G_{XTC} has girth 4, that is, the shortest cycle in G_{XTC} is of length 4.*

Proof. It is sufficient to show that G_{XTC} does not contain any cycles of length 3. We suppose for contradiction that there exists such a cycle through the nodes u, v, and w, that is, all three edges (u, v), (v, w), and (u, w) are contained in G_{XTC}. Let us further assume without loss of generality that $w \prec_u v$, or $\omega_{uw} < \omega_{uv}$. At the point of time when u considered the inclusion of v in N_u, w had already been processed. This means that—since (u, v) is in G_{XTC}—$u \prec_v w$ must hold, and consequently also $\omega_{uv} < \omega_{vw}$. Applying the same argument from v's perspective yields that also $v \prec_w u$ or $\omega_{vw} < \omega_{uw}$ must hold, which provokes a contradiction. □

Although G_{XTC} has girth 4, it does not feature sparseness: Constructed on a general weighted graph G, G_{XTC} can have degree $\Theta(n)$ and contain $\Theta(n^2)$ edges. An example for such a graph is $K_{n/2,n/2}$, the complete bipartite graph with $n/2$ nodes in each partition set, in which each node has degree $n/2$ and

which consequently contains $n^2/4$ edges in total. Presumably, network graphs resulting from real ad hoc networks will however be considerably sparser.

14.5 Average-Case Evaluation

In this section we will present the properties of the topology control graph G_{XTC} on average-case Euclidean graphs, that is on graphs generated by randomly and uniformly placing nodes in a given field. In particular, we will study the spanner and bounded degree properties—Properties 2+ and 3+ as stated in the introduction of this chapter—in the context of average graphs. The bounded degree property having been shown to hold for G_{XTC} in Section 14.3, we will demonstrate in this section that also the spanner property holds on average graphs. Since G_{XTC}, being a planar graph, lends itself to geographic routing, we will furthermore examine the influence of G_{XTC} on such routing algorithms.

In order to model the physical network in our average-case evaluation, we will adopt the unit disk graph definition, in which an edge exists if and only if its Euclidean length is less than one unit. To assess the average-case properties of G_{XTC}, we will compare it with the Gabriel Graph [38]. Being one of the most prominent topology control structures on the one hand, the Gabriel Graph is on the other hand particularly well suited for comparison with respect to the spanner property since it is not only a spanner (with constant stretch factor) with respect to the energy metric, but even contains the energy-minimal path between any pair of nodes.

As already described in Chapter 3, the Gabriel Graph is—similarly to the Relative Neighborhood Graph—defined such that the presence of an edge (u, v) depends on whether a certain geometric region contains a third node w or not. In the case of the Gabriel Graph this geometric region is the disk (including its boundary) having the line segment $\overline{uv}$ as a diameter (cf. Figure 14.4(b) opposing the Gabriel Graph definition to the Relative Neighborhood Graph).

Figure 14.5 illustrates the Gabriel Graph and G_{XTC} constructed from a sample unit disk graph. The figure shows well that, informally speaking, areas with high edge density in the unit disk graph are thinned out by topology control while connectivity of the graph is preserved. This tendency can be observed even more clearly for G_{XTC} than for the Gabriel Graph.

As studied in percolation theory and also addressed in Section 8.1, *network density* is an important parameter influencing the properties of average-case networks. The transition between the two extremes with respect to network density—very low densities, where hardly any pair of nodes is connected, and very high densities, where disconnection of the network is extremely improbable—takes place in a relatively narrow *critical density range* roughly around 5 nodes per unit disk. In order to account for this effect, we acquired our measurements and simulation results over a spectrum of network densities. For each considered network density, the number of nodes corresponding to the density was randomly and uniformly placed on a square field with side length 20 units.

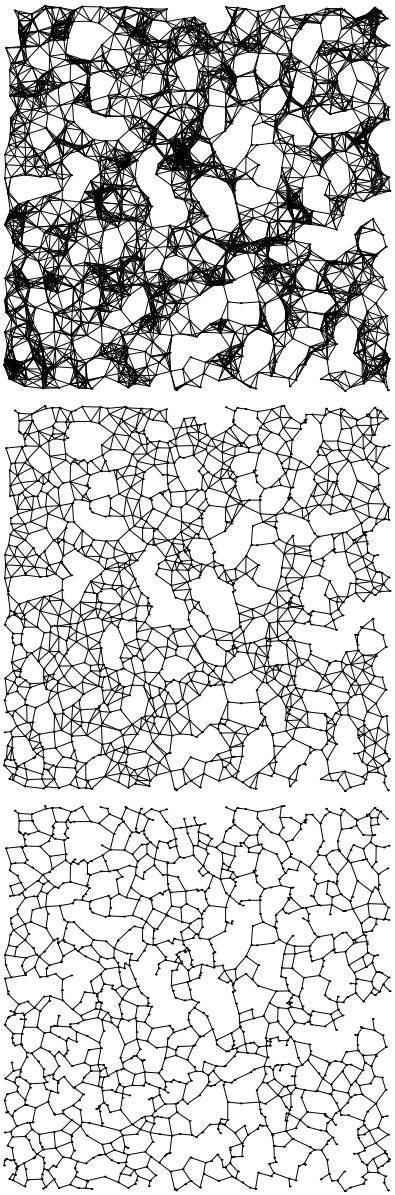

Figure 14.5: The unit disk graph G (top), the Gabriel Graph of G (center), and G_{XTC} of 1400 nodes placed randomly and uniformly on a square field of 20 units side length.

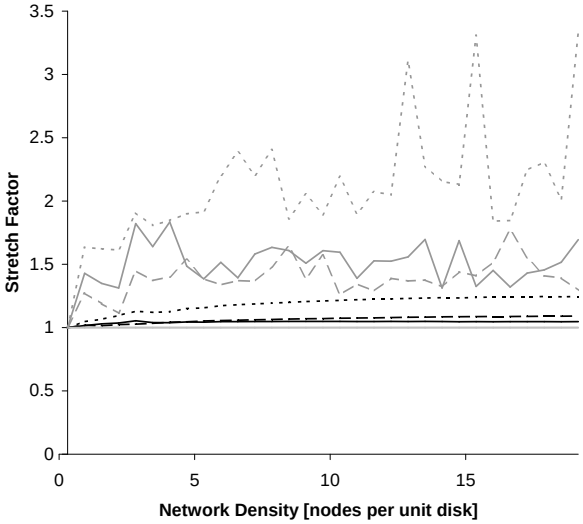

Figure 14.6: Stretch factors of G_{XTC} with respect to the energy (solid) and the Euclidean (dotted) metric as well as the stretch factor of the Gabriel Graph with respect to the Euclidean metric (dashed), all over a network density spectrum. Mean values are plotted in black, maximum values in gray. The Gabriel Graph contains the energy optimal path; its stretch factor curve ($\equiv 1$, light gray) is plotted for reference.

14.5.1 Spanner Property

In order to study the spanner property of G_{XTC} on randomly generated networks, we calculated the stretch factor of a pair of nodes u, v

$$s(u,v) := \frac{c(p^*_{tc}(u,v))}{c(p^*(u,v))},$$

that is the ratio between the cost of the shortest path between u and v on the topology control graph and of the shortest path on the unit disk graph G. For each considered network density, we generated 2000 networks and also randomly selected a pair of nodes u, v to calculate $s(u,v)$. G_{XTC} and the Gabriel Graph were employed as topology control graphs. Edge and correspondingly path cost measures were considered with respect to Euclidean edge length $c_d(\cdot)$ and to energy $c_E(\cdot)$ with an attenuation exponent 2.

Figure 14.6 depicts our results over the considered network density range. Since the Gabriel Graph contains the energy-minimal path connecting any pair of nodes, its stretch factor with respect to the energy metric $s(u,v) \equiv 1$ is

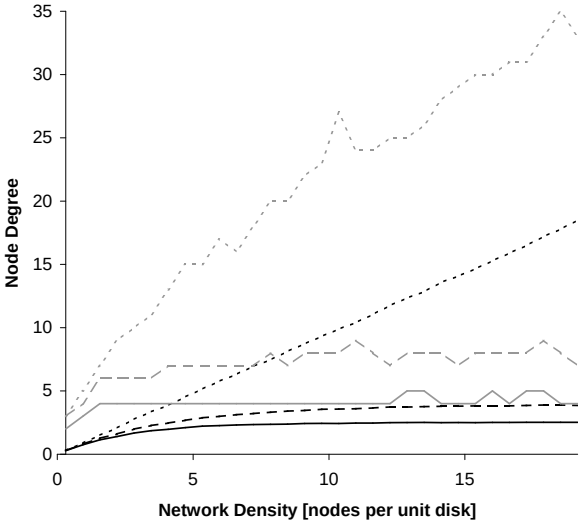

Figure 14.7: Node degree of G_{XTC} (solid), the Gabriel Graph (dashed), and the unit disk graph (dotted) over network density spectrum. Mean values are plotted in black, maximum values in gray.

plotted for reference. Although G_{XTC} is not an energy spanner with constant stretch in a strict sense, the results show that this graph has a good energy-spanning property on average graphs: The corresponding mean values of the stretch factor do not exceed 1.06, while even the maximum values stay below 1.9. With respect to the Euclidean metric, the mean value curves for both the Gabriel Graph and G_{XTC} remain—throughout the density range—almost constant at low values of below 1.1 and 1.25, respectively; the corresponding maximum curve for G_{XTC} is less stable than for the Gabriel Graph, but only rarely reaches values above 3.

In summary, the results show that G_{XTC} is a good average-case spanner with respect to the Euclidean metric, but especially for the energy metric.

14.5.2 Bounded Degree Property

G_{XTC} being shown to have degree at most 6 in Theorem 14.3, we will study here its average-case behavior with respect to its node degree by comparison with the corresponding behavior of the Gabriel Graph. The results for this comparison were obtained similarly as for the spanner property. From each of the 2000 random networks—generated for each considered network density—, a

randomly chosen node was examined regarding its degree in the unit disk graph, the Gabriel Graph G, and G_{XTC}.

Figure 14.7 shows the computed results. Mean and maximum degree values for the unit disk graph rise in accordance with network density. The mean degree curves for G_{XTC} and the Gabriel Graph increase slowly for very low densities, remain however—once beyond the critical density range—constant at approximately 2.5 and 3.9, respectively. Although inherently less stable, the maximum degree numbers for G_{XTC} and the Gabriel Graph stay within a narrow range from 4 to 5 and between 7 and 9. The low degree values of G_{XTC} suggest its suitability to reduce interference in average-case ad hoc networks.

14.5.3 Performance of Geographic Routing

G_{XTC}'s planarity property enables it to be employed for geographic routing. We will study its influence on geographic routing again by comparison with the Gabriel Graph.

For this purpose, the results were obtained by simulation of the GOAFR$^+$ algorithm (cf. Chapter 7) on G_{XTC} and the Gabriel Graph of 2000 randomly generated networks for each considered network density. As in Section 8.3, the performance measure for the algorithm A routing from a source s to a destination t on a network G (s and t having been chosen randomly from G) is defined as

$$cost_A(G, s, t) := \frac{s_A(G, s, t)}{c_\ell(p^*_\ell(G, s, t))},$$

that is the number of steps taken by the algorithm normalized by the hop length of the shortest path from s to t on G.

Figure 14.8 shows the typical bell-shaped characteristic of the mean algorithm cost curves around the critical density range (cf. Chapter 8). The critical network density range is additionally identified by the sharp increase of the network connectivity rate, that is the frequency with which a randomly chosen node pair at a given network density is connected. The figure depicts that above all in the critical density range the performance of the GOAFR$^+$ algorithm on G_{XTC} is slightly worse than on the Gabriel Graph. In light of the cost curve of the well-known GFG/GPSR routing algorithm (see Section 8.3), however, the degradation of GOAFR$^+$ employed on G_{XTC} compared to the Gabriel Graph can be considered negligible.

14.6 Concluding Remarks

In this chapter we introduced the XTC topology control algorithm. Compared to previous proposals for topology control, XTC has three main advantages. First, it is not only simple, but also local: Every node communicates with its neighbors in the network not more than twice. Second, unlike many other topology control algorithms, XTC does not require the network graph to be a

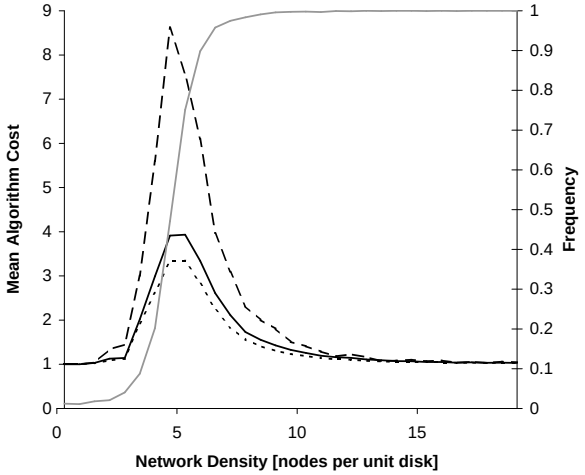

Figure 14.8: Mean cost of the GOAFR$^+$ routing algorithm on G_{XTC} (solid), on the Gabriel Graph (dotted), and of the GFG/GPSR algorithm on the Gabriel Graph (dashed). For reference, the network connectivity rate is plotted against the right y axis.

Euclidean Graph, let alone a unit disk graph. Also for the global case of the network graph being a general weighted graph, XTC proves correct and computes a resulting subgraph maintaining connectivity. Third, while previously proposed topology control algorithms commonly assumed that exact node and neighbor position information is available, XTC does not require this assumption. The algorithm works with the general notion of a quality order over a node's neighbors. Whereas correctness of the algorithm can be shown even without any strict assumptions on this neighbor order, the resulting topology features the bounded degree property provided that the neighbor order corresponds to Euclidean distances and that the network is a unit disk graph. On average-case random unit disk graphs, the resulting graph also shows good spanner properties, above all with respect to the energy metric. Being planar, the proposed topology is finally also suitable for geographic routing.

In this introductory topology control chapter, we discussed a lightweight topology control algorithm which tries to operate independently of unrealistic assumptions. The XTC algorithm can be considered to follow the example of "classic" topology control in that it provably features several beneficial properties. In the next and subsequent chapters we will go beyond this approach and focus explicitly on one fundamental issue in wireless networks, often maintained to be solved or at least tackled implicitly by previously proposed topology control structures: interference.

Chapter 15

Does Topology Control Reduce Interference?

I have never let my schooling interfere with my education.
Mark Twain (1835–1910)

As already mentioned in the introduction of this second part of the dissertation, one very general definition of topology control is to—given a communication network—construct a subgraph with certain beneficial properties. The previous chapter forms an example of a topology control algorithm following these lines, while specially considering omission of unrealistic assumptions and simplicity of construction.

In this and the subsequent chapters we will focus on an equally popular conception of topology control: Very simply put, the main goal of topology control is often understood to be reduction of energy consumed by the network nodes in order to extend network lifetime. Since the amount of energy required to transmit a message increases at least quadratically with distance, it makes sense to replace a long link by a sequence of short links. On the one hand, energy can therefore be conserved by abandoning energy-expensive long-range connections, thereby allowing the nodes to reduce their transmission power levels. On the other hand, confining transmission ranges also reduces interference, which in turn lowers node energy consumption by reducing the number of collisions and consequently packet retransmissions on the media access layer. Dropping communication links however clearly takes place at the cost of network connectivity: If too many edges are abandoned, connecting paths can grow unacceptably long or the network can even become completely disconnected. As illustrated in Figure 15.1, topology control can therefore be considered a trade-off between energy conservation and interference reduction on the one hand and connectivity on the other hand.

Figure 15.1: Topology control constitutes a trade-off between node energy conservation and network connectivity.

The interference aspect is often maintained by developers of topology control algorithms to be solved by sparseness or low node degree of the resulting topology graph, without providing rigorous justification or proofs. The foremost contribution of this chapter is to disprove this assertion.

In contrast to most of the related work—where the interference issue is seemingly solved by sparseness arguments—, we will start out by precisely defining our notion of interference. This definition of interference is based on the natural question of how many nodes are affected by communication over a certain link. By prohibiting specific network edges, the potential for communication over high-interference links can then be confined.

We will employ this interference definition to formulate the trade-off between energy conservation and network connectivity. In particular we will state certain requirements that need to be met by the resulting topology. Among these requirements are connectivity (if two nodes are—possibly indirectly—connected in the given network, they should also be connected in the resulting topology) and the constant-stretch spanner property (the shortest path between any pair of nodes on the resulting topology should be longer at most by a constant factor than the shortest path connecting the same pair of nodes in the given network). After stating such requirements, an optimization problem can be formulated to find the topology meeting the given requirements with minimum interference.

For the requirement that the resulting topology retain connectivity of the given network, we will show that most of the currently proposed topology control algorithms—already by having every node connect to its nearest neighbor—commit a substantial mistake: Although certain proposed topologies are guaranteed to have low degree yielding a sparse graph, interference becomes asymptotically incomparable with the interference-minimal topology. We will also show that there exist graphs for which no local algorithm can approximate the optimum. With respect to the sometimes desirable requirement that the resulting topology should be planar, we will show that planarity can increase interference.

Furthermore we will propose a centralized algorithm (LIFE) that computes an interference-minimal connectivity-preserving topology. For the requirement that the resulting topology be a spanner with a given stretch factor, we will present (based on a centralized variant of the algorithm) a distributed local algorithm (LLISE) that computes a provably interference-optimal spanner topology.

Our results are not confined to worst-case considerations; we will also show by simulation that on average-case graphs traditional topology control algorithms—in particular the Gabriel Graph and the Relative Neighborhood Graph—fail to effectively reduce interference. Moreover, we will show that these constructions are outperformed by the LLISE algorithm, which therefore appears average-case effective in addition to its worst-case optimality.

In the following section we will introduce the interference model discussed in this chapter. Focusing on the drawbacks of currently proposed topology control algorithms with respect to interference in Section 15.2, we will present interference-optimal algorithms in the subsequent section. Section 15.4 will assess our algorithms as well as previously proposed topologies regarding their interference on average-case graphs.

15.1 Model

As in several previous places in this dissertation, an ad hoc network is modeled as a graph $G = (V, E)$ consisting of a set of nodes $V \subset \mathbb{R}^2$ in the Euclidean plane and a set of edges $E \subseteq V^2$. Nodes represent mobile hosts, whereas edges stand for links between nodes. In order to prevent already basic communication between directly neighboring nodes from becoming unacceptably cumbersome [92], it is required that a message sent over a link can be acknowledged by sending a corresponding message over the same link in the opposite direction. In other words, only *undirected* (symmetric) edges are considered.

We assume that a node can adjust its transmission power to any value between zero and its maximum power level. The maximum power levels are not assumed to be equal for all nodes. An edge (u, v) may exist only if both incident nodes are capable of sending a message over (u, v), in particular if the maximum transmission radius of both u and v is at least $|uv|$, their Euclidean distance. A pair of nodes u, v is considered *connectable in the given network* if there exists a path connecting u and v provided that all transmission radii are set to their respective maximum values. The task of a *topology control* algorithm is then to compute a subgraph of the given network graph with certain properties, reducing the transmission power levels and thereby attempting to lower interference and energy consumption.

With a chosen transmission radius—for instance to reach a node v—a node u affects at least all nodes located within the circle centered at u and with radius $|uv|$. $D(u, r)$ denoting the disk centered at node u with radius r and requiring edge symmetry, we consequently define the *coverage* of an (undirected) edge

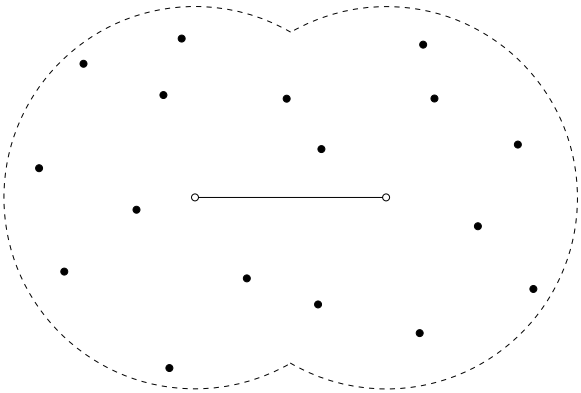

Figure 15.2: Nodes covered by a communication link.

$e = (u, v)$ to be the cardinality of the set of nodes covered by the disks[1] induced by u and v:

$$Cov(e) := \big|\{w \in V | w \text{ is covered by } D(u, |uv|)\} \cup \{w \in V | w \text{ is covered by } D(v, |vu|)\}\big|.$$

In other words, the coverage $Cov(e)$ represents the number of network nodes affected by nodes u and v communicating with their transmission powers chosen such that they exactly reach each other (cf. Figure 15.2).

The edge level interference defined so far is now extended to a graph interference measure as the maximum coverage occurring in a graph:

Definition 15.1. *The interference of a graph G=(V,E) is defined as*

$$I(G) := \max_{e \in E} Cov(e).$$

Since interference reduction per se would be senseless (if all nodes simply set their transmission power to zero, interference will be reduced to a minimum), the formulation of additional requirements to be met by a resulting topology is necessary. A resulting topology can for instance be required

- to maintain connectivity of the given communication graph (if a pair of nodes is connectable in the given network, it should also be connected in the resulting topology graph),

[1]The results of this chapter can also be adapted to the case where transmission ranges are not perfect circles centered at the sending node. We adhere to this simplified model for clarity of representation.

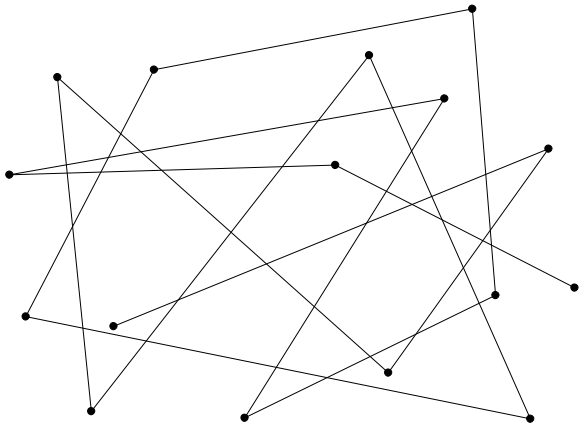

Figure 15.3: Low degree does not guarantee low interference.

- to be a spanner with constant stretch of the underlying graph (the shortest path connecting a pair of nodes u, v on the resulting topology is longer by a constant factor only than the shortest path between u and v on the given network), or
- to be planar (no two edges in the resulting graph intersect).

Finding a resulting topology which meets one or a combination of such requirements with minimum interference constitutes an optimization problem.

15.2 Interference in Known Topologies

It is often argued that sparse topologies with small or bounded degree are well suited to minimize interference. In this section we will show that low degree does not necessarily imply low interference. Moreover, we will demonstrate that most of the currently known topology control algorithms can perform badly compared to the interference optimum, that is a topology which minimizes interference in the first place.

In particular, we will consider in this section the basic problem of constructing an interference-minimal topology maintaining connectivity of the given network.

The following basic observation states that—although often maintained—low degree alone does not guarantee low interference. Figure 15.3, for instance, shows a topology graph with degree 2 whose interference is however roughly n, the number of network nodes. A node can interfere with other nodes that are not direct neighbors in the chosen topology graph. Whereas the maximum degree of the underlying communication graph of the given network (with all

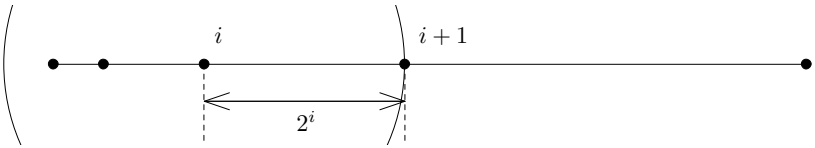

Figure 15.4: Exponential node chain with interference $\Omega(n)$.

nodes transmitting at full power) is an upper bound for interference, the degree of a resulting topology graph is only a lower bound.

There exist instances where also the optimum exhibits interference $\Omega(n)$, for instance a chain of nodes with exponentially growing distances (cf. Figure 15.4, proposed in [74]), whose large interference is caused as a consequence of the requirement that the resulting topology is to be connected. Every node u_i (except for the leftmost) is required to have an incident edge, which covers all nodes left of u_i. Assessing the interference quality of a topology control algorithm therefore implies its interference on a given network needs to be compared to the optimum interference topology for the same network.

To the best of our knowledge, all currently known topology control algorithms constructing only symmetric connections have in common that every node establishes a symmetric connection to at least its nearest neighbor. In other words, all these topologies contain the Nearest Neighbor Forest constructed on the given network. In the following we will show that owing to the inclusion of the Nearest Neighbor Forest as a subgraph, the interference of a resulting topology can become incomparably bad with respect to a topology with optimum interference.

Theorem 15.1. *No currently proposed topology control algorithm establishing only symmetric connections—required to maintain connectivity of the given network—is guaranteed to yield a nontrivial interference approximation of the optimum solution. In particular, interference of any proposed topology can be* $\Omega(n)$ *times larger than the interference of the optimum connected topology, where* n *is the total number of network nodes.*

Proof. Figure 15.5 depicts an extension of the example graph shown in Figure 15.4. In addition to a horizontal exponential node chain, each of these nodes h_i has a corresponding node v_i vertically displaced by a little more than h_i's distance to its left neighbor. If d_i denotes this vertical distance, the inequality $d_i > 2^{i-1}$ holds. These additional nodes form a second (diagonal) exponential line. Between two of these diagonal nodes v_{i-1} and v_i, an additional helper node t_i is placed such that $|h_i, t_i| > |h_i, v_i|$.

The Nearest Neighbor Forest for this given network (with the additional assumption that the transmission radius of each node can be chosen sufficiently

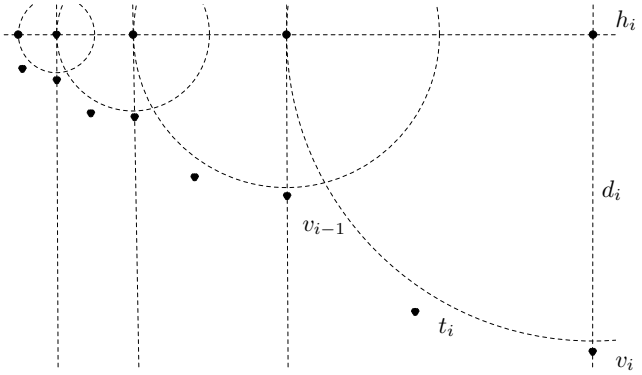

Figure 15.5: Two exponential node chains.

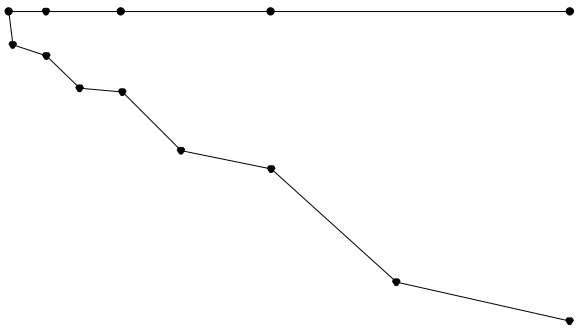

Figure 15.6: The Nearest Neighbor Forest yields interference $\Omega(n)$.

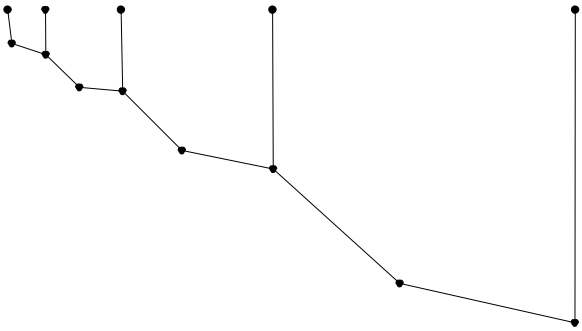

Figure 15.7: Optimal tree with constant interference.

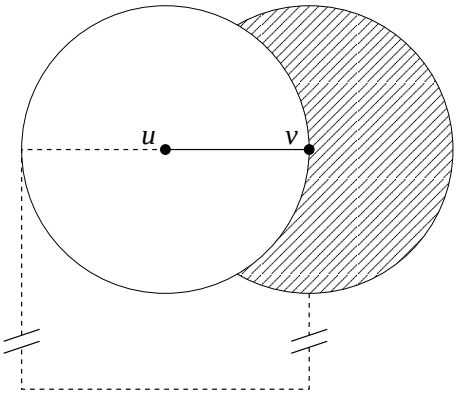

Figure 15.8: Worst-case graph for which no local algorithm can approximate optimum interference.

large) is shown in Figure 15.6. Roughly one third of all nodes being part of the horizontally connected exponential chain, interference of any topology containing the Nearest Neighbor Forest amounts to at least $\Omega(n)$. An interference-optimal topology, however, would connect the nodes as depicted in Figure 15.7 with constant interference. □

In other words, already by having each node connect to the nearest neighbor, a topology control algorithm makes an "irrevocable" error. Moreover, it commits an asymptotically worst possible error since the interference in any network cannot become larger than n.

As roughly one third of all nodes are part of the horizontal exponential node chain in Figure 15.5, the observation stated in Theorem 15.1 would also hold for an average interference measure, averaging interference over all edges.

The following theorem even shows that for connectivity-preserving topologies no local algorithm can approximate optimum interference for every given network. Thereby the definition of a distributed *local* algorithm assumes that each network node is informed about its network neighborhood only up to a given constant distance.

Theorem 15.2. *For the requirement of maintaining connectivity of the given network, there exists a class of graphs for which no local algorithm can approximate optimum interference.*

Proof. In Figure 15.8 the maximum transmission radius of a node is $|uv|$. n being the number of nodes in the graph, let the shaded area contain $\Omega(n)$ evenly distributed nodes which can be connected with constant interference. For each such node w, the inequalities $|wv| < |uv|$ and $|uw| > |uv|$ hold. It

follows that the edge (u, v) has interference in $\Omega(n)$ since it covers all nodes in the shaded area. In addition, there is a chain of nodes (dashed path) connecting node u with node v indirectly through the nodes located in the shaded area. The nodes in the chain are located in such a way that it is possible to connect them with constant interference. For such a graph, O(1) interference can be achieved by connecting u to the rest of the graph through the chain of nodes and not directly through edge (u, v), which would cause $\Omega(n)$ interference.

A local algorithm at node u has to decide if it can drop edge (u, v). This is only possible if u knows about the existence of an alternative path from u to v in order to maintain connectivity. By elongating the chain sufficiently, the local algorithm can thus be forced to include edge (u, v), pushing up interference to $O(n)$ whereas the optimum is $\Omega(1)$. □

As mentioned in Section 15.1, another popular requirement for topology control algorithms besides bounded degree is planarity of the resulting topology, meaning that no two edges of the resulting graph intersect. This is often desired for the application of geographic routing algorithms that are only applicable to planar graphs. But topology control algorithms enforcing planarity are not optimal in terms of interference:

Theorem 15.3. *There exist graphs on which interference-optimal topologies—required to maintain connectivity—are not planar.*

Proof. In Figure 15.9, the maximum transmission radius of a node is $|ab|$. All eligible edges are depicted together with the coverage areas for the edges whose incident nodes are both in $\{a, b, c, d\}$. The indicated weight of an edge e corresponds to its coverage $Cov(e)$. V and W represent sets of 3 and 4 nodes, respectively, which can be connected among themselves with interference 3. A topology control algorithm can only reduce interference by removing all edges with maximum interference (here (a, c) and (b, c)) from the graph. Thereafter, no further edge can be removed without breaking connectivity since the graph without (a, c) and (b, c) is a tree. Thus, the resulting tree is interference-optimal and non-planar, as both edges (a, b) and (c, d) have to remain in the resulting topology. □

15.3 Low-Interference Topologies

In this section we will present three algorithms that explicitly reduce interference of a given network. The first algorithm is capable of finding an interference-optimal topology maintaining connectivity of the given network. The other two algorithms compute an interference-optimal topology under the additional requirement of constructing a spanner of the given network. Whereas the first spanner algorithm assumes global knowledge of the network, the second can be computed locally.

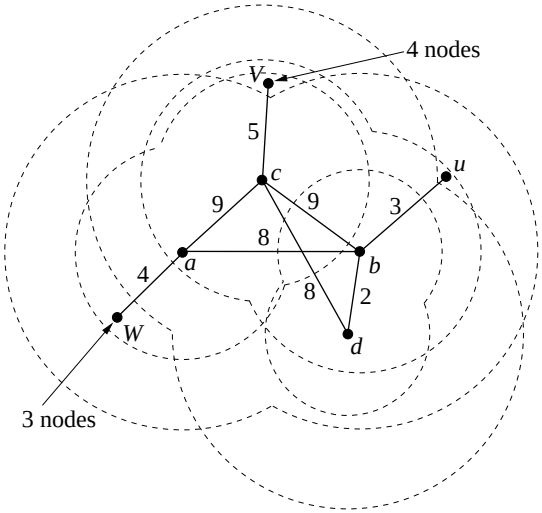

Figure 15.9: Node set whose interference-optimal topology is not planar.

15.3.1 Interference-Optimal Spanning Forest

In the following we again require the resulting topology to maintain connectivity of the given network. A topology graph meeting this requirement can therefore consist of a tree for each connected component of the given network since additional edges do not contribute to graph connectivity while potentially unnecessarily increasing interference. A *Minimum Interference Forest* is therefore a set of trees maintaining the connectivity of the given network with least possible interference. As the following theorem shows, the LIFE algorithm computes such a forest.

Theorem 15.4. *The forest constructed by LIFE is a Minimum Interference Forest.*

Proof. The LIFE algorithm computes a minimum spanning forest (MSF) of the graph $G = (V, E)$, where E is the set of all eligible edges, if every edge $e \in E$ is attributed the weight $Cov(e)$. With its greedy strategy, it follows the lines of Kruskal's MSF algorithm [26]. To prove the theorem, it is therefore sufficient to show that the MSF is optimal with respect to interference. Optimality of LIFE then follows from the fact that a minimum spanning forest also minimizes the maximum edge weight in any spanning forest. (Assuming for contradiction that G^* is an MSF with maximum weight edge e^*, whereas G_{LIFE} is a spanning forest with lower maximum edge weight, e^* could be replaced by a corresponding

Low Interference Forest Establisher (LIFE)

Input: a set of nodes V, each $v \in V$ having attributed a maximum transmission radius r_v^{max}

1: E = all eligible edges (u, v) ($r_u^{max} \geq |uv|$ and $r_v^{max} \geq |uv|$)
   // E will contain all unprocessed edges
2: $E_{LIFE} = \emptyset$
3: $G_{LIFE} = (V, E_{LIFE})$
4: **while** $E \neq \emptyset$ **do**
5: $e = (u, v) \in E$ with minimum coverage
6: **if** u, v are not connected in G_{LIFE} **then**
7: $E_{LIFE} = E_{LIFE} \cup \{e\}$
8: **end if**
9: $E = E \setminus \{e\}$
10: **end while**

Output: Graph G_{LIFE}

edge from G_{LIFE}, yielding a spanning forest with total edge weight smaller than G^*'s, which contradicts the assumption that G^* is an MSF.) □

With an appropriate implementation of the connectivity query in Line 6, the running time of the algorithm LIFE is $\mathrm{O}(n^2 \log n)$. If the given network is known to consist of only one connected component, Prim's minimum-spanning-tree algorithm can be employed with running time $\mathrm{O}(n^2)$. Algorithms computing a minimum spanning tree in a distributed way—as particularly suitable for ad hoc networks—are described in detail in [86].

15.3.2 Low-Interference Spanners

LIFE optimizes interference for the requirement that the resulting topology has to maintain connectivity. In addition to connectivity it is often desired that the resulting topology should be a spanner with constant stretch of the given network. A spanner with stretch factor t can be formally defined as follows:

Definition 15.2 (t-Spanner). *A t-spanner of a graph $G = (V, E)$ is a subgraph $G' = (V, E')$ such that for each pair (u, v) of nodes $c(p^*_{G'}(u, v)) \leq t \cdot c(p^*_G(u, v))$, where $c(p^*_{G'}(u, v))$ and $c(p^*_G(u, v))$ denote the length of the shortest path between u and v in G' and G, respectively.*

In this chapter we consider Euclidean spanners, that is, the length of a path is defined as the sum of the Euclidean lengths of all its edges (cf. Chapter 3). With slight modifications, our results are however also extendable to hop spanners, where the length of a path corresponds to the number of its edges.

Algorithm LISE is a topology control algorithm that constructs a t-spanner with optimum interference. LISE starts with a graph $G_{LISE} = (V, E_{LISE})$ where E_{LISE} is initially the empty set. It processes all eligible edges of the

Low Interference Spanner Establisher (LISE)

Input: a set of nodes V, each $v \in V$ having attributed a maximum transmission radius r_v^{max}; a stretch factor $t \geq 1$

1: E = all eligible edges (u, v) ($r_u^{max} \geq |uv|$ and $r_v^{max} \geq |uv|$)
   // E will contain all unprocessed edges
2: $E_{LISE} = \emptyset$
3: $G_{LISE} = (V, E_{LISE})$
4: **while** $E \neq \emptyset$ **do**
5: $e = (u, v) \in E$ with maximum coverage
6: **while** $c(p^*(u, v)$ in $G_{LISE}) > t\,|uv|$ **do**
7: f = edge $\in E$ with minimum coverage
8: move all edges $\in E$ with coverage $Cov(f)$ to E_{LISE}
9: **end while**
10: $E = E \setminus \{e\}$
11: **end while**

Output: Graph G_{LISE}

given network $G = (V, E)$ in descending order of their coverage. For each edge $(u, v) \in E$ not already in E_{LISE}, LISE checks whether there exists a path from u to v in G_{LISE} with Euclidean length at most $t\,|uv|$. As long as no such path exists, the algorithm keeps inserting all unprocessed eligible edges with minimum coverage into E_{LISE}.

To prove the interference optimality of G_{LISE}, we introduce an additional lemma, which shows that G_{LISE} contains all eligible edges whose coverage is less than $I(G_{LISE})$.

Lemma 15.5. *The graph $G_{LISE} = (V, E_{LISE})$ constructed by LISE from a given network $G = (V, E)$ contains all edges e in E whose coverage $Cov(e)$ is less than $I(G_{LISE})$.*

Proof. We assume for the sake of contradiction that there exists an edge e in E with $Cov(e) < I(G_{LISE})$ which is not contained in E_{LISE}. Consequently, LISE never takes an edge with coverage $Cov(e)$ in Line 7 since the algorithm would insert all edges with $Cov(e)$ into E_{LISE} in Line 8 instantly (thus also e). There exists however an edge f in E_{LISE} with $Cov(f) = I(G_{LISE})$ eventually taken in Line 7. Therefore the inequality $Cov(e) < Cov(f)$ holds. At the time the algorithm takes f in Line 7, all edges taken in Line 5 must have had coverage greater than or equal to $Cov(f)$ since the maximum of an ordered set can only be greater than or equal to the minimum of the same set. Hence e has never been taken in Line 5 and therefore has never been removed from E in Line 10. Consequently, e is still in E when f is taken as the edge with minimum coverage in E. Thus $Cov(f) \leq Cov(e)$ holds, which leads to a contradiction. □

With Lemma 15.5 we are ready to prove that the resulting topology constructed by LISE is an interference-optimal t-spanner.

Theorem 15.6. *The graph $G_{LISE} = (V, E_{LISE})$ constructed by LISE from a given network $G = (V, E)$ is an interference-optimal t-spanner of G.*

Proof. In order to show that G_{LISE} meets the spanner property, it is sufficient to prove that for each edge $(u, v) \in E$ there exists a path in G_{LISE} with length not greater than $t\,|uv|$. This holds since for a shortest path $p^*(u, v)$ in G a path $p'(u, v)$ in G_{LISE} with $c(p') \leq t\,c(p)$ can be constructed by substituting for each edge on p the corresponding spanner path in G_{LISE}. For edges in E which also occur in E_{LISE}, the spanner property is trivially true. On the other hand an edge (u, v) can only be in E but not in E_{LISE} if a path from u to v in G_{LISE} with length not greater than $t\,|uv|$ exists (see if-condition in Line 6). Thus, G_{LISE} is a t-spanner of G.

Interference optimality of LISE can be proved by contradiction. We therefor assume, that G_{LISE} is not an interference-optimal t-spanner. Let $G^* = (V, E^*)$ be an interference-optimal t-spanner for G. Since G_{LISE} is not optimal, it follows that $I(G_{LISE}) > I(G^*)$. Thus all edges in E^* have coverage strictly less than $I(G_{LISE})$. It follows from Lemma 15.5 that E^* is a nontrivial subset of E_{LISE}. Let T be the set of edges in E_{LISE} with coverage $I(G_{LISE})$ and let $\tilde{G} = (V, \tilde{E})$ be the graph with $\tilde{E} = E_{LISE} \setminus T$. $\tilde{G}$ is a t-spanner since E^* is still a subset of $\tilde{E}$, and $I(\tilde{G}) \leq I(G_{LISE}) - 1$ holds. Because T is eventually inserted into E_{LISE} in Line 8, there exists an edge $(u, v) \in E$ that was taken in Line 5 and for which no path $p(u, v)$ exists in $\tilde{G}$ with $c(p) \leq t\,|uv|$. Thus, $\tilde{G}$ is no t-spanner (and therefore neither G^*), which contradicts the assumption that G^* is an interference-optimal t-spanner. □

As regards the running time of LISE, it computes for each edge at most one shortest path. This holds since multiple shortest path computations for the same edge in Line 6 cause at least as many edges to be inserted into E_{LISE} in Line 8 without computing shortest paths for them. Since finding a shortest alternative path for an edge requires $\mathrm{O}(n^2)$ time and as the network contains at most the same amount of edges, the overall running time of LISE is as well polynomial in the number of network nodes.

In contrast to the problem of finding a connected topology with optimum interference, the problem of finding an interference-optimal t-spanner is locally solvable. The reason for this is that finding an interference-optimal path $p(u, v)$ for an edge (u, v) with $c(p) \leq t\,|uv|$ can be restricted to a certain neighborhood of (u, v).

In the following we will describe a local algorithm similar to LISE that is executed at all eligible edges of the given network. In reality, algorithm LLISE (Local LISE) is executed for each edge by one of its incident nodes (for instance the one with the greater identifier). The description of LLISE assumes the point of view of an edge $e = (u, v)$. The algorithm consists of three main steps:

1) Collect $(\frac{t}{2})$-neighborhood,

2) compute minimum interference path for e, and

LLISE

1: collect $(\frac{t}{2})$-neighborhood $G_N = (V_N, E_N)$ of $G = (V, E)$
2: $E' = \emptyset$
3: $G' = (V_N, E')$
4: **repeat**
5: f = edge $\in E_N$ with minimum coverage
6: move all edges $\in E_N$ with coverage $Cov(f)$ to E'
7: p = shortestPath$(u - v)$ in G'
8: **until** $c(p) \leq t\,|uv|$
9: inform all edges on p to remain in the resulting topology.

Note: $G_{LL} = (V, E_{LL})$ consists of all edges eventually informed to remain in the resulting topology.

3) inform all edges on that path to remain in the resulting topology.

In the first step, e gains knowledge of its $(\frac{t}{2})$-neighborhood. For a Euclidean spanner, the k-neighborhood of e is defined as all edges that can be reached (or more precisely at least one of their incident nodes) over a path p starting at u or v, respectively, with $c(p) \leq k\,c(e)$. Knowledge of the $(\frac{t}{2})$-neighborhood at all edges can be achieved by local flooding (cf. Chapter 10).

Theorem 15.7. *The graph $G_{LL} = (V, E_{LL})$ constructed by LLISE from a given network $G = (V, E)$ is an interference-optimal t-spanner of G.*

During the second step, a minimum-interference path p from u to v with $c(p) \leq t\,c(e)$ is computed. LLISE starts with a graph $G_{LL} = (V, E_{LL})$ consisting of all nodes in the $(\frac{t}{2})$-neighborhood and an initially empty edge set. It inserts edges consecutively into E_{LL}—in ascending order according to their coverage—, until a shortest path $p^*(u, v)$ is found in G_{LL} with $c(p^*) \leq t\,c(e)$.

In the third step, e informs all edges on the path found in the second step to remain in the resulting topology. The resulting topology then consists of all edges receiving a corresponding message. In the following we will show that it is sufficient for e to limit the search for an interference-optimal path $p(u, v)$ meeting the spanner property to the $(\frac{t}{2})$-neighborhood of e.

Lemma 15.8. *Given an edge $e = (u, v)$, no path p from u to v with $c(p) \leq t\,c(e)$ contains an edge which is not in the $(\frac{t}{2})$-neighborhood of e.*

Proof. For the sake of contradiction we assume that a path p from u to v with $c(p) \leq t\,c(e)$ contains at least one edge (w, x) not in the $(\frac{t}{2})$-neighborhood of e. Without loss of generality we further assume that, traversing p from u to v, we visit w before x. Since (w, x) is not in the $(\frac{t}{2})$-neighborhood, by definition, no path from u to w with length less than or equal to $(\frac{t}{2})|e|$ exists (the same holds for any path from v to x). Consequently, the inequality $c(p) > t\,c(e) + |wx|$ holds, which contradicts the assumption that $c(p) \leq t\,c(e)$. □

With Lemma 15.8 we are now able to prove that the topology constructed by LLISE is a t-spanner with optimum interference.

Proof. The spanner property of LLISE can be proved similar to the first part of the proof of Theorem 15.6, where LISE is shown to be a t-spanner.

In order to show interference optimality, it is sufficient to prove that the spanner path constructed for any edge $e = (u, v) \in G$ by LLISE is interference-optimal, where interference of a path is defined as the maximum interference of an edge on that path. The reason for this is that only edges that lie on one of these paths remain in the resulting topology; non-optimality of G_{LL} would therefore imply non-optimality of at least one of these spanner paths. In the following we look at the algorithm executed by $e = (u, v)$. In Line 6, edges in E are consecutively inserted into E', starting with $E' = \emptyset$, until a spanner path p from u to v is found in Line 8. Since LLISE inserts the edges into E' in ascending order according to their coverage and since p is the first path meeting the spanner property, p is an interference-optimal t-spanner path from u to v in the $(\frac{t}{2})$-neighborhood. From Lemma 15.8 we know that the $(\frac{t}{2})$-neighborhood of e contains all spanner paths from u to v and therefore also the interference-optimal one. Thus it is not possible that LLISE does not see the global interference-optimal t-spanner path due to its local knowledge about G. Consequently, p is the global interference-optimal t-spanner path of e. □

15.4 Average-Case Interference

In this section we will consider interference of topology control algorithms in average-case graphs, that is in graphs with randomly placed nodes.

In particular, networks were constructed by placing nodes randomly and uniformly on a square field of size 20 by 20 units and subsequently computing for each node set the unit disk graph (cf. Definition 3.1). The resulting unit disk graphs were then employed as input networks for topology control. Since node density is a fundamental property of networks with randomly placed nodes, the networks were generated over a spectrum of node densities.

15.4.1 Connectivity-Preserving Topologies

To evaluate connectivity-preserving topologies on average-case graphs, two well-known topology control structures are considered, in particular the Gabriel Graph [38] and the Relative Neighborhood Graph [102]. The interference-reducing effect of these two constructions is considered by comparison with the interference value of the given unit disk graph network on the one hand and with the interference-optimal connectivity-preserving topology on the other hand. The interference-optimal topology was constructed by means of the LIFE algorithm presented in Section 15.3.

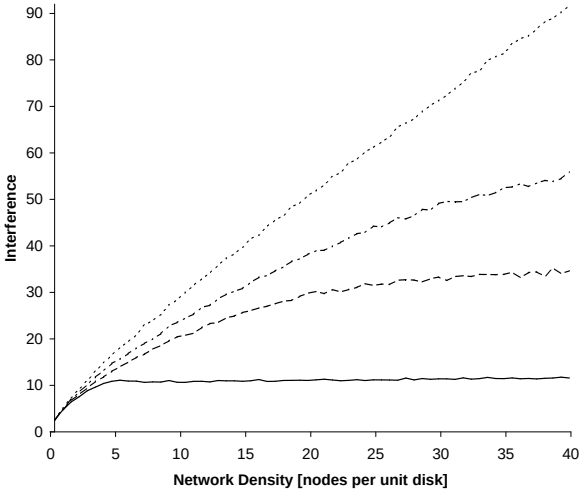

Figure 15.10: Interference values of the unit disk graph without topology control (dotted), the Gabriel Graph (dash-dotted), the Relative Neighborhood Graph (dashed), and the interference-optimal connectivity-preserving topology (solid).

Figure 15.10 shows the interference mean values over 1000 networks for each simulated network density. While the resulting interference curves behave similarly for very low network densities, they fall into three groups with increasing density: At a density of roughly 5 network nodes per unit disk, the interference-optimal curve levels off and remains at a value of approximately 11.5. On the other hand the interference curve of the unit disk graph without topology control rises almost linearly. Between these two extremes the Gabriel Graph and Relative Neighborhood Graph values increase clearly more slowly than the unit disk graph curve, but show significantly higher values than the interference-optimal topology.

The simulation results show that the edge reduction performed by the Gabriel Graph and Relative Neighborhood Graph constructions reduce interference of the given network; this effect is clearer with the Relative Neighborhood Graph due to its stricter edge inclusion criterion and consequently its being a subgraph of the Gabriel Graph (see also Figure 14.4). However, the interference values of these two constructions are considerably higher than the results of the interference-optimal connectivity-preserving topology. Furthermore, although (unless in special cases) the Relative Neighborhood Graph has degree at most 6, it is not even clear whether with increasing network density the respective interference curve remains around the maximum value found so far or whether it would increase further for densities beyond the simulated spectrum. It can

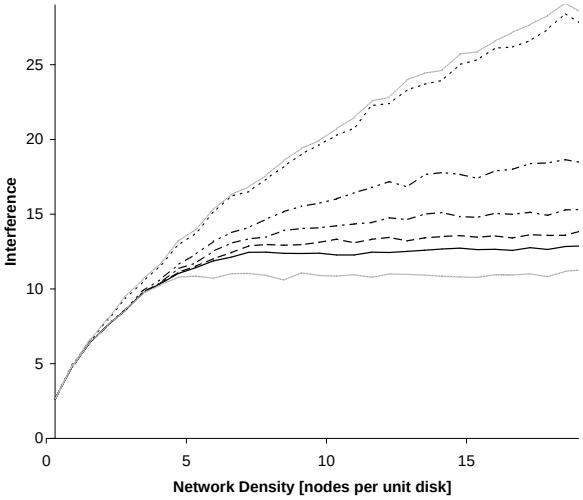

Figure 15.11: Interference values of LLISE for stretch factors 2 (dotted), 4 (dash-dot-dotted), 6 (dash-dotted), 8 (dashed), and 10 (solid). Interference values of the Relative Neighborhood Graph (upper gray) and the interference-optimal connectivity-preserving topology (lower gray) are plotted for reference.

therefore be concluded that also for average-case graphs sparseness does not imply low interference.

15.4.2 Low Interference Spanners

Going beyond connectivity-preserving topologies, we will consider in this section constant-stretch spanners, that is topologies guaranteeing that the shortest paths on the resulting topology are only by a constant factor longer than on the given network (cf. Section 15.3.2).

Figure 15.11 depicts simulation results—in particular the mean interference values over 100 networks at each simulated network density—of the topology constructed by the LLISE algorithm introduced in Section 15.3 for different stretch factors t. The simulation results show that by increasing the requested stretch factor it is possible to achieve interference values close to the optimum interference values caused by connectivity-preserving topologies as described in the previous section. Moreover, even with a low stretch factor of 2, LLISE does not perform worse than the Relative Neighborhood Graph, which is *not* a constant-stretch spanner. In summary, the simulation results show that the LLISE algorithm performs well with respect to interference also on average-case graphs. An illustration of the simulation graphs is provided in Figure 15.12.

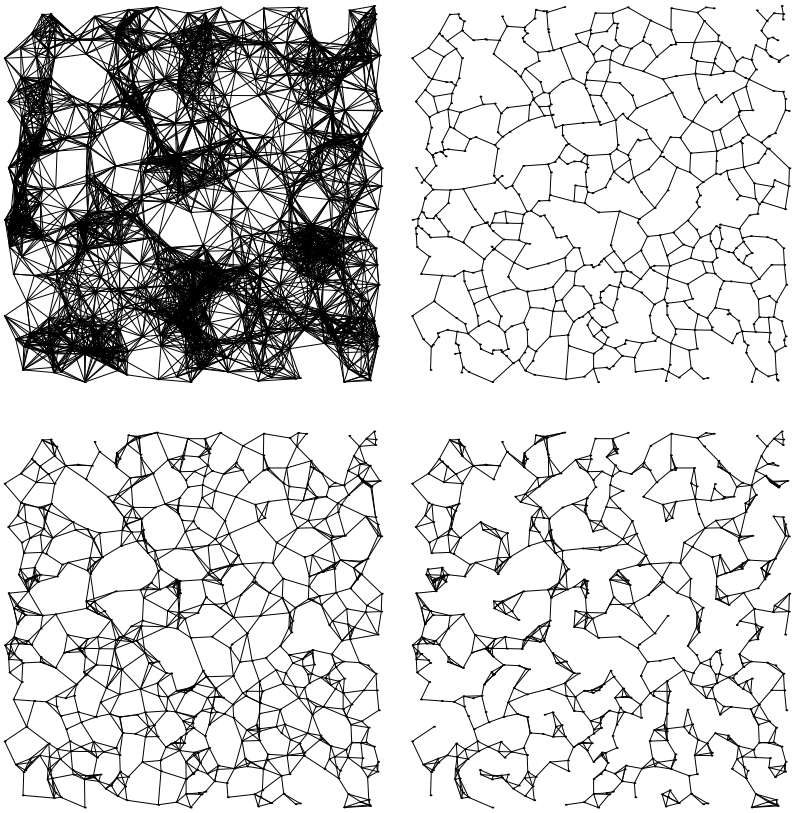

Figure 15.12: The unit disk graph G (top left, interference 50), the Relative Neighborhood Graph of G (top right, interference 25), G_{LL} computed by LLISE with stretch factors 2 (bottom left, interference 23) and 10 (bottom right, interference 12) at a network density of 20 nodes per unit disk on a square field of 10 units side length. Note that, for instance in the western region of the graph, LLISE—depending on the chosen stretch factor—omits high-interference "bridge" edges if alternative spanning paths exist.

15.5 Concluding Remarks

In this chapter we disprove the widely advocated assumption that sparse topologies automatically imply low interference. In contrast to most of the related work we provide an intuitive definition of interference. With this interference model we show that currently proposed topology control constructions—although claiming so—do not in the first place focus on reducing interference.

In addition, we propose provably interference-minimal connectivity-preserving and spanner constructions. A locally computable version of the interference-minimal spanner construction can even be considered practicable since it is shown to significantly outperform previously suggested topology control algorithms also on average-case graphs.

As important as an explicit definition of interference is for its analysis, a clear drawback of the definition presented in this chapter is that it assumes the perspective of the *sender* of a message. Formally, this notion is reflected by the definition of the coverage of an edge, which counts the number of network nodes affected by communication over the considered edge. This stands in opposition to the fact that signal disturbance and message collisions actually occur at the intended *receiver* of a message. This characteristic of interference forms the core of the following chapters discussing approaches to model interference which assume the signal receiver's perspective.

Chapter 16

Receiver-Centric Interference in Sensor Networks

When a book and a head collide and there is a hollow sound,
is it always from the book?
Georg Christoph Lichtenberg (1742–1799)

The previous chapter states two main points. First, an implicit notion of interference—as advocated by most of the previous work—can lead to topology control algorithms that fail to effectively reduce interference. Second, it introduces an explicit definition of interference, based on the number of nodes potentially disturbed by communication over a link.

In contrast to this approach, we will assume in this chapter a receiver-centric perspective. Particularly, we will formulate an interference definition at the heart of which lies the question by how many other nodes a given network node can be disturbed. Compared to the sender-centric interference definition proposed in the previous chapter, the definition of interference presented in this chapter reflects intuition more closely in the sense that interference is considered at the receiver, where message collisions prevent proper reception. Informally, our interference definition can also be considered to correspond to the effort required to avoid collisions, be it by means of time division multiplexing—assigning transmission time slots such that no two messages collide at a receiving node—, by means of frequency division multiplexing—having messages sent in different assigned frequency bands—, or by means of code division multiplexing—where small interference allows for reduced coding overhead.

In this chapter, we will consider interference in sensor networks. A sensor network consists of sensors deployed in a given region with the task of sensing a

certain physical value (such as temperature, humidity, brightness, or motion). The sensors are equipped with radio devices and—in the popular monitoring scenario model—periodically transfer the sensed data to a designated data sink node. To allow all data to be gathered at the sink, a topology control algorithm therefore constructs a *sink tree*, a directed tree with all arcs (directed edges)—modeling unidirectional communication links—pointing towards the sink node. In the context of interference reduction, the task of the topology control algorithm is to find such a sink tree with least possible interference. Thereby we account for the fact that in the monitoring scenario, communication from the sink to the sensors occurs rarely and can therefore be neglected with respect to interference.

Assuming a worst-case perspective, we will show that there are network instances in which any topology control algorithm will construct a resulting network with interference at least $\log n - 1$. We furthermore propose the *Nearest Component Connector (NCC)* algorithm, which provably produces at most $O(\log n)$ interference in any network in polynomial time. In this sense, the NCC algorithm is asymptotically optimal. In a second part of this chapter we will compare the NCC algorithm with previously proposed structures in average networks. On the one hand we will show that—besides being asymptotically worst-case optimal—NCC also in the average case produces interference results comparable with previously proposed structures. On the other hand, a minimum-spanning-tree-based structure not originally designed to reduce interference interestingly appears to outperform NCC in average-case networks, while it cannot guarantee to produce low interference in worst-case examples.

The chapter is organized as follows: We will first introduce our receiver-centric interference model and a formulation of the considered interference minimization problem in the following section. While Section 16.2 shows that there exist network instances in which any topology control algorithm will produce interference at least $\log n - 1$, the subsequent section presents the NCC algorithm and proves that it matches this lower bound. Section 16.4 finally discusses interference generated by NCC and other algorithms in average-case networks.

16.1 Model and Notation

In this section we will describe our model of a sensor network and formally define interference and interference minimization in the context of our model.

Our model of a sensor network is a directed graph $G(V, E)$ where nodes $v_1, \ldots, v_n$ placed in the plane represent the set of sensors including the sink. Communication links between sensors are modeled as (directed) arcs. We assume that the transmission power of each node can be adjusted. Higher transmission power allows a node to send messages over a longer distance. We assume that the covered area of a sending node v_i is a disk with v_i in its center. Furthermore we assume that a node can reach another node only if it is at most 1 distance unit away. In other words, the graph consisting of all eligible

arcs if all nodes set their transmission power to the maximum possible values corresponds to the unit disk graph constructed given the node set V (cf. Definition 3.1). Finally we only consider *connectable* graphs, which means that—with all transmission radii set to their maximum values—a path from any node to any other node in the network is constructible or—more technically—the unit disk graph given the node set V consists of one connected component.

If we want to minimize interference in sensor networks, we have to look at topologies in which each node sends its data to at most one other node, and a valid graph contains a path from every sensor to the sink. These two requirements result in a tree with the sink as its root and all arcs pointing towards the root. We call such a tree a *sink tree*. Figure 16.1 shows a sample sink tree with 6 nodes.

Definition 16.1. *Given a set of nodes V and a sink s, a sink tree is a tree spanning V with all arcs pointing towards s.*

We use an explicit model of interference. We explicitly count the number of nodes potentially disturbing reception of a message. This definition reflects the fact that interference is a problem occurring at the receiver. Minimizing the interference at each possible receiver (each node in the network) reduces the number of potential message collisions in the network and therefore lowers the probability of required retransmission. This approach intends to save energy and extend the lifetime of sensors equipped with batteries.

In particular, the interference value of a single node is defined to be the number of transmission circles by which the node is covered.

Definition 16.2. *The interference value of a single node v is defined as*

$$I(v) := |\{u | u \neq v \wedge v \in D(u, r_u)\}|$$

where $D(u, r_u)$ stands for the transmission circle with node u in its center and radius r_u.

The interference of a whole network is defined as the maximum of all interference values in the graph (see Figure 16.1).[1]

Definition 16.3. *The interference of a Graph $G(V, E)$ is defined as*

$$I(G) := \max_{v \in V} I(v).$$

The problem we study in this chapter consists in finding a sink tree with least possible interference for a given sensor network.

[1] It can be argued similarly that the interference of a whole network can be defined as the *average* of the node interference values. Such a definition is not considered in this chapter. Note that with this alternative definition of interference, the problem of finding a valid data-gathering structure with minimum interference can be solved optimally constructing a Minimum Directed Spanning Tree with arc weights corresponding to the number of nodes covered by each edge.

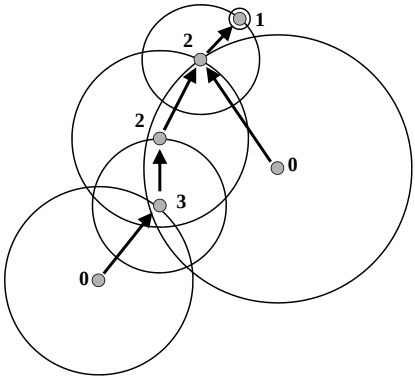

Figure 16.1: A sink tree with 6 nodes, the uppermost node being the sink node. Each node is labeled with its interference value. The interference of the whole network is 3.

Definition 16.4. *The Minimum-Interference Sink Tree (MIST) problem is defined as the problem of finding a sink tree for a given node set with minimal interference.*

In the remainder of this chapter we will consider topology control algorithms with the goal of solving the MIST problem.

16.2 A Lower Bound

In this section we will show that n nodes in a sensor network can be arranged in a way that there exists no algorithm able to construct a sink tree with interference less than $\log(n) - 1$. The existence of such examples constitutes a lower bound with respect to interference.

Theorem 16.1. *There exist sensor networks with nodes arranged in a way that no algorithm can construct a sink tree with interference less than* $\log(n) - 1$.

Proof. To prove this theorem, we present an arrangement of $n = 2^s$ nodes which cannot be connected to the given sink in the described way with interference less than $\log(n) - 1$. The nodes are arranged on a horizontal line, as illustrated in Figure 16.2. The first $k = 4$ nodes v_1, v_2, v_3, and v_4, are positioned at coordinates 0, $1{\cdot}\nu$, $3{\cdot}\nu$, and $4{\cdot}\nu$, where $\nu = 1/(4{\cdot}3^{s-2})$ is a normalization factor. Then a copy of the already positioned nodes is placed in distance $d = |v_1 v_k|$ to the right of node v_k. This construction is recursively repeated until 2^s nodes are placed on the horizontal line. Note that due to normalization of the node positions every node can reach any other node in the network.

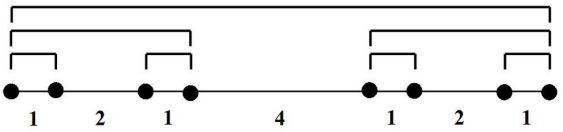

Figure 16.2: A recursive arrangement of 16 nodes on a horizontal line. The labels indicate distances without normalization by the factor $1/(4 \cdot 3^{s-2})$.

After the execution of any possible algorithm, there has to exist a directed path from each node in the set to the global sink. Let G_1 be the node group $\{v_1, \ldots, v_{2^{s-1}}\}$ and $G_2 := \{v_{2^{s-1}+1}, \ldots, v_{2^s}\}$. If we assume without loss of generality that the node group G_2 contains the global sink, the result of an algorithm has to contain an arc from G_1 to G_2. According to the special arrangement of the nodes in our example, the gap between G_1 and G_2 has length equal to the (Euclidean) diameter of the two groups. The arc between G_1 and G_2 cannot be shorter than this gap and therefore interferes with all nodes but one in G_1. Figure 16.3 illustrates the idea of the proof.

If, in a next step, we look into G_1, there are 2^{s-1} nodes partitioned into two subgroups $G_{1.1}$ and $G_{1.2}$. Assuming, again without loss of generality, that the above arc from G_1 to G_2 originates in $G_{1.2}$, we can observe that there must exist an arc leading out from $G_{1.1}$, which—bridging $G_{1.1}$'s adjacent gap—interferes with all nodes in $G_{1.1}$ (except for the node at which this arc originates). The existence of such an arc is a consequence of the required directed path from each node to the sink. The same argument recursively holds for all node levels in the arrangement.

This all together proves that at least one node is affected on every level of recursion. And as the node set is of size 2^s, we obtain a maximum interference of at least $s - 1$ or $\log(n) - 1$. □

16.3 NCC Algorithm

Having presented a lower bound on interference in the previous section, we will introduce in this section the Nearest Component Connector algorithm *(NCC)* matching this lower bound and being described in detail in Algorithms 16.1 and 16.2.

The general idea of this algorithm is to connect components to their nearest neighbors. This is done in several rounds and leads to a sink tree. A component can be a single node or a group of previously connected nodes. When the algorithm starts, each node in the given sensor network forms a component of its own. First, the predefined global sink is treated exactly as a normal node. Whenever two or more components are connected in one round, they form

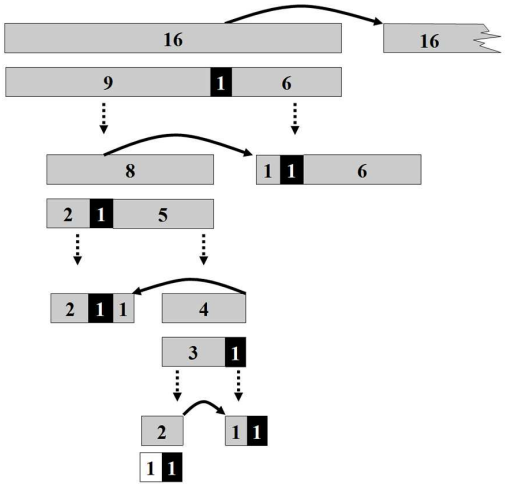

Figure 16.3: Illustration of the proof of Theorem 16.1. The numbers stand for the number of nodes in each group. The dark fields represent the unaffected nodes in the last step and the hollow node (bottom) is the one whose interference value is incremented in each step.

a single component in the following round of NCC. Considering an arbitrary component at any point of time of the algorithm execution, we observe that this component has exactly one node all other component members have a directed path to. This means that there is one node which gathers all sensed data of the component. We call this node the *local sink* of its component.

Whenever a new arc is established during the execution of NCC, it goes from a local sink of a component C to the nearest node not in C. However, due to the fact that all nodes have maximum transmission range 1, it is possible that the current sink s of a component C cannot connect to any node outside C. In this case another node s' is designated to become the new sink of C, particularly the nearest node to s (with respect to the number of hops) capable of reaching any node outside C. This is accomplished by removing all arcs originating at nodes on the shortest path $p^*(s, s')$ from s to s' and subsequently adding the arcs along $p^*(s, s')$ (cf. Algorithm 16.2). Note that every component contains at least one node capable of reaching another node outside its component since we only consider connectable networks.

If a round—connecting every sink to its nearest neighbor outside its component—produces a cycle, this cycle is broken by removing one of its arcs at the end of the round. This leads to the construction of a valid sink tree topology. It is possible that, after the last round of NCC, the root of the resulting tree is not necessarily the global sink. In this case, the root of the resulting tree is moved

Input: V: a set of nodes placed in the plane
$s_g \in V$: a predefined global sink
1: $G := (V, E := \emptyset)$
2: $lsinks := V$ // set of local sinks
3: **while** $|lsinks| > 1$ **do**
4: **for all** $s \in lsinks$ **do**
5: $E' := \emptyset$
6: $C :=$ component containing s
7: **if** s cannot reach any node outside C **then**
8: $s' :=$ nearest node to s (hop metric) capable of reaching a node outside C
9: $movesink(G, s, s')$
10: $s := s'$
11: **end if**
12: $E' := E' \cup \{e\}$, where e is the arc from s to its nearest neighbor (Euclidean distance) outside C
13: **end for**
14: **if** $G' := (V, E \cup E')$ contains cycles **then**
15: remove one of the arcs in each cycle from E'
16: **end if**
17: $G := G'$
18: $lsinks :=$ sinks in G // sinks are nodes having no outgoing arc
19: **end while**
20: $s :=$ only remaining sink in $lsinks$
21: **if** $s \neq s_g$ **then**
22: $movesink(G, s, s_g)$
23: **end if**
Output: G

Algorithm 16.1: Nearest Component Connector Algorithm NCC

Input: Graph $G = (V, E)$
s_1: a local sink in G
s_2: a node in the same component as s_1
1: $sp :=$ shortest path from s_1 to s_2 according to the hop metric
2: remove all arcs originating at nodes on sp (including s_2) from E
3: add arcs on sp to E

Algorithm 16.2: Procedure $movesink(G, s_1, s_2)$

to the global sink again by means of the *movesink* procedure (Algorithm 16.2). Figure 16.4 shows a sample execution of the NCC algorithm.

We will now prove that the presented NCC algorithm constructs a valid sink tree topology for a given sensor network consisting of n nodes with an

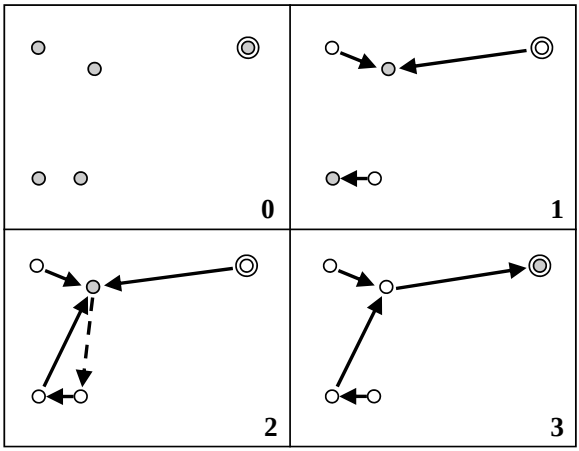

Figure 16.4: A sample execution of NCC on a given set of 5 nodes. Situation 0 shows the given nodes and the predefined sink (top right node) In each of the following two rounds, every local sink connects to the nearest node not in its own component. In Round 2, a cycle is produced. It is broken at the end of the round by removing one of the involved arcs (dashed arrow). After the last round (Situation 3) the arc originating from the global sink is removed and an arc is added from the only remaining local sink to the predefined global sink. For clarity of representation, the node distances are assumed to be sufficiently small such that execution of the *movesink* procedure is not required.

interference value in $O(\log n)$. We will also see that the execution of NCC takes polynomial time only.

Theorem 16.2. *The NCC Algorithm constructs a sink tree on a given Graph* $G = (V, E)$ *with* $|V| = n$ *producing an interference value of at most* $O(\log n)$ *in polynomial time.*

Proof. This proof has three parts. In the first part we show that NCC does not need more than $\log n$ rounds (while-loop iterations) to build the sink tree. In the second part we show that in each of these rounds the interference value of a node will not be incremented by more than a constant value. In part three we show that NCC terminates in time polynomial in the total number of nodes.

To show the first part, we use the fact that in each round a local sink s either establishes an arc to the nearest node of another component or that another component establishes an arc to the component s is part of. The two or more connected components together form one component in the next round. Therefore the number of components in round i is at most half the number of

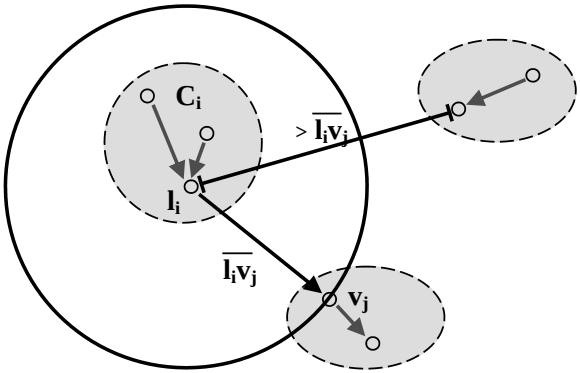

Figure 16.5: An illustration of the second part of the proof of Theorem 16.2. An arc from the local sink l_i to v_j only interferes with nodes in l_i's component and v_j (and nodes in other components only if they are exactly at distance $\overline{l_i v_j}$).

the components in round $i-1$. This implies that after at most $\log n$ rounds only one component is left and the algorithm terminates.

In the second part of the proof we show that each round increases the interference value at most by a constant. First, the *movesink*(G, s, s') procedure only increases the interference values of nodes in C, the component of s and s', since none of the nodes on the shortest path from s to s' can reach any node outside C; furthermore, introduction of the arcs on the shortest path from s to s' increases the interference value by at most 3. (A node v suffering interference increase of at least 4 would contradict the fact that only arcs on a shortest path are added to the graph; the shortest path could be shortcut via v, exploiting that in the unit disk graph model all arcs of length up to one unit are eligible.) Second, once the sink of a component C can reach another node outside C, we use the fact that each sink connects to the *nearest* node in a component different from its own, as illustrated in Figure 16.5. If a local sink l_i being contained in component C_i connects to a node v_j, its distances from all nodes not in C_i are at least $|l_i v_j|$. Therefore only nodes which are members of component C_i or nodes with the same distance from l_i as v_j are affected by the new arc. Furthermore, a component establishes at most one new arc in a single round and maximally 6 local sinks can establish an arc to the same node.[2] All this shows that the interference value of a node is incremented at most by a constant in a single round of NCC.

[2]This is the so-called "kissing number." It is defined as the number of equivalent spheres that touch an equivalent sphere without intersections. The kissing number in the two-dimensional plane is 6. In three dimensions it is 12.

Together with part one of the proof and the fact that the *movesink* procedure is applied at most one more time, it follows that the interference value of any node is incremented at most $\log(n) + 1$ times and each time by at most a constant value. This proves that the interference of the whole network is in $O(\log n)$.

The remaining part of Theorem 16.2 to be proved states that NCC terminates in polynomial time. Every node v_i is a sink in one iteration of the while loop. (Actually it can be a sink in more than one loop iteration if a cycle is broken by removing the arc originating at that node. The fact however that for every such removed arc at least one other arc is added to the tree in the same round entails only an additional factor 2.) A sink has to find its nearest neighbor in a foreign component. This can be implemented using a list of neighbors, sorted according to their distances for each node and a union-find structure to check if a node is in a foreign component. The n lists of sorted neighbors can be constructed in time $O(n \cdot n \log(n))$. Maintaining the union-find structure and component membership lookups during the execution of NCC can also be performed in time $O\big(n^2 \log(n)\big)$, while the total cost for shortest path computation is in $O\big(n \cdot n^2\big)$. These observations prove that NCC terminates in polynomial time. □

We present NCC in a centralized manner. This reflects the fact that in a sensor network we have an instance (the sink) commonly assumed to have much more computing power and energy than all other nodes (sensors). Therefore the sink can run NCC and distribute the topology information of the constructed sink tree in an initialization phase.

Nevertheless a distributed variant of NCC without the coordination of a central instance is feasible. This variant would require counters in each node which keep track of the number of component unions the node has been involved in since the start of the algorithm. These counters then guarantee that only components in the same "round" can establish new arcs between each other. Also computation of shortest paths and the *movesink* procedure are implementable in a distributed way using a variant of flooding and by sending according messages over the shortest path thereby found.

16.4 Interference in Average-Case Networks

We have so far seen that the NCC algorithm has asymptotically optimal worst-case behavior in the sense that it produces interference not greater than $O(\log n)$ for all possible node arrangements. In this section we will have a closer look at the average-case behavior of our algorithm. We do this by simulation. The nodes in our simulations are distributed randomly and uniformly in a square field. Also the sink is chosen randomly. To see how our algorithm behaves in average-case networks, we compare it to two other construction methods for sink trees which have been proposed previously as data gathering structures.

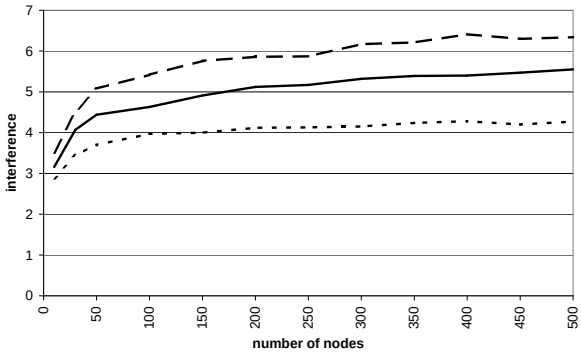

Figure 16.6: Simulation results for the Shortest Path Tree algorithm (dashed), the Nearest Component Connector algorithm (solid) and the Minimum Spanning Tree algorithm (dotted) over network sizes ranging from 10 to 500 nodes.

These two methods are:

1. The Minimum Spanning Tree algorithm (MST) with weights equal to the Euclidean edge lengths, all edges pointing towards the global sink.

2. The Shortest Path Tree algorithm (SPT) with respect to the energy metric $c_E(\cdot)$ (cf. Chapter 3). (The SPT contains the shortest paths from all nodes to the sink.) All edges point towards the global sink.

In order to allow for evaluation in different conditions, all three algorithms constructed sink trees for networks with different node densities. We simulated networks from 10 to 500 nodes distributed in a square with its side length chosen such that all nodes are mutually visible (in order to emphasize the differences between the simulated algorithms without restricting the set of eligible arcs by introduction of the unit disk graph model). Plotted in the diagram are the averaged values over 100 runs for each simulated node density.

Figure 16.6 shows that all three algorithms produce increasing interference with rising node densities. Closer observation yields that our NCC algorithm performs better than the Shortest Path Tree algorithm but worse than the Minimum Spanning Tree algorithm. This is quite intriguing, as the very simple MST algorithm, which was not explicitly designed to reduce interference, seems to outperform our NCC algorithm in average-case networks. Note however that the MST is not asymptotically worst-case optimal and can produce interference of $n - 2$ for a sensor network consisting of n nodes. A sample of such an arrangement is shown in Figure 16.7.

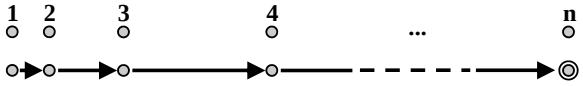

Figure 16.7: Nodes arranged on a horizontal line with exponentially increasing distances, the rightmost node chosen as the global sink. Applied to this setting, the MST algorithm produces interference $n - 2$.

16.5 Concluding Remarks

The approach we assume in this chapter in order to study interference in wireless and particularly sensor networks differs from most of the previous work in two ways: First, we introduce an explicit definition of interference. Second, in contrast to the interference model presented in the previous chapter, our definition of interference is receiver-centric and reflects the fact that message collisions prevent proper message reception only if they occur at the receiving node.

With this formalized notion of interference, we show on the one hand that there exist instances of sensor networks with n nodes in which it is impossible to construct a sink tree—a valid data gathering structure—with interference less than $\log n - 1$. On the other hand, we describe the NCC algorithm asymptotically matching this lower bound in that it provably builds a sink tree with interference at most $O(\log n)$ on any given sensor network. In addition to these worst-case observations, we also evaluate the NCC algorithm in average networks. Intriguingly, the latter results show that—although the interference values produced by NCC fall roughly in the same range as those of other constructions—a simple minimum-spanning-tree-based structure keeps interference at a lower level than NCC in average-case networks.

The considerations presented in this chapter are restricted to sensor networks. Technically, this is reflected in the formulation of the Minimum-Interference Sink Tree problem. In the following chapter we will endeavor a first step towards extending a receiver-centric approach to the modeling of interference in more general ad hoc networks, particularly dropping the requirement of constructing a sink tree and instead focusing on graph connectivity.

Chapter 17

A Robust Interference Model for Ad Hoc Networks

Thanks to the Interstate Highway System,
it is now possible to travel from coast to coast without seeing anything.
Charles Kuralt (1934–1997)

The definition of interference introduced in Chapter 15 is problematic in two respects. First, it is based on the number of nodes affected by communication over a given link. In other words, interference is considered to be an issue at the sender instead of at the receiver, where message collisions actually prevent proper reception. It can therefore be argued that such sender-centric perspective hardly reflects real-world interference. Chapter 16 presents an approach to the modeling of interference from a receiver-centric perspective in the context of sensor networks.

The second weakness of the model introduced in Chapter 15 is of more technical nature. According to its definition of interference, adding a single node to a given network can dramatically influence the interference measure. In the network depicted in Figure 17.1, addition of the rightmost node to the cluster of roughly homogeneously distributed nodes entails the construction of a communication link covering all nodes in the network; accordingly—merely by introduction of one additional node—the interference value of the represented topology is pushed up from a small constant to the maximum possible value, that is the number of nodes in the network. This behavior contrasts to the intuition that a single additional node also represents but one additional packet source potentially causing collisions.

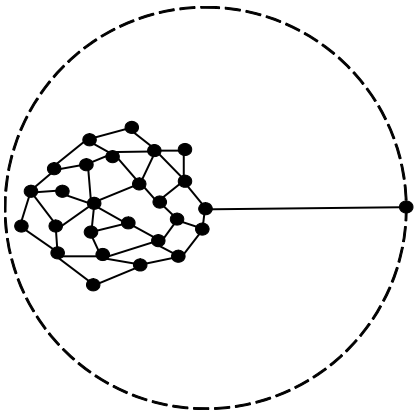

Figure 17.1: In the interference model presented in Chapter 15, addition of a single node increases interference from a small constant to the maximum possible value, the total number of network nodes.

In contrast to this sender-centric interference definition, we will adopt in this chapter the model presented in the previous chapter, explicitly considering interference at its point of impact, particularly at the receiver. Informally, the definition of interference considered in this chapter is based on the natural question by how many other nodes a given network node can be disturbed. In this chapter we will adapt the concept of interference—introduced in Chapter 16 for sensor networks—to be suitable also in general ad hoc networks. Technically, we will consider arbitrary undirected networks as opposed to the directed data gathering trees studied in Chapter 16.

Interestingly, our interference definition not only reflects intuition due to its receiver-centricity. This definition also results in a robust interference model in terms of measure increase due to the arrival of additional nodes in the network. Particularly, an additional node causes an interference increase of at most one at other nodes of the network. In clear contrast to the sender-centric model from Chapter 15, this corresponds to reality, where one added node contending for the shared medium constitutes only one additional possible collision source for nearby nodes in the network.

As already mentioned earlier, interference reduction as an end of its own is meaningless—every node setting its transmission power to a minimum value trivially minimizes interference—without the formulation of additional requirements to be met by the resulting topology. In this chapter we will study the fundamental requirement that the considered topology control algorithms preserve connectivity of the given network. Similar as in Chapter 15, we will show that for this requirement most of the currently proposed topology control algorithms trying to implicitly reduce interference commit a substantial mistake—

even by having every node connect to its nearest neighbor. Based on the intuition that already one-dimensional networks exhibit most of the complexity of finding minimum-interference topologies, we will precisely anatomize networks restricted to one dimension—a model also known as the *highway* model. We will first look at a particular network where distances between nodes increase exponentially from left to right. [74] introduces this network as a high interference example yielding interference $O(\Delta)$, where Δ is the maximum node degree. We will show that it is intriguingly possible to achieve interference $O(\sqrt{\Delta})$ in our model for this network, which matches a lower bound also presented in this chapter. Based on the insights thereby gained, we will then consider general highway instances where nodes can be distributed arbitrarily in one dimension. For the problem of finding a minimum-interference topology while maintaining connectivity, we will propose an approximation algorithm with approximation ratio $O(\sqrt[4]{\Delta})$.

The chapter is organized as follows: Section 17.1 formally introducing the model studied in this chapter, Section 17.2 will focus on the drawbacks of currently proposed topology control algorithms with respect to interference. In the subsequent section we will consider the important case where nodes are distributed in one dimension by providing a lower bound for the interference in such networks and presenting an algorithm that matches this lower bound.

17.1 Network and Interference Model

Also in this chapter we model the wireless network with the unit disk graph G, where Δ refers to the maximum node degree in G. Again, in order to prevent already basic communication between neighboring nodes from becoming unacceptably cumbersome, we require that a message sent over a link can be acknowledged by sending a corresponding message over the same link in the opposite direction. In other words, only *undirected* (symmetric) edges are considered.

We assume that each node can adjust its transmission power to any value between zero and its maximum transmission power level. The main goal of a *topology control* algorithm is then to compute a subgraph of the given network graph G that maintains connectivity even if reducing transmission power levels of the nodes in V and thereby attempting to reduce interference and energy consumption.

Let N_u denote the set of all neighbors of a node $u \in V$ in the resulting topology. Then, each node u features a value r_u defined as the distance from u to its farthest neighbor. More precisely $r_u = max_{v \in N_u} |uv|$, where $|uv|$ denotes the Euclidean distance between nodes u and v. Since we assume the nodes to use omnidirectional antennas, $D(u, r_u)$ denotes the disk centered at u with radius r_u covering all nodes that are possibly affected by message transmission of u to one of its neighbors. The transmission radii of the network nodes having been fixed, the definition of interference corresponds to the definition in the

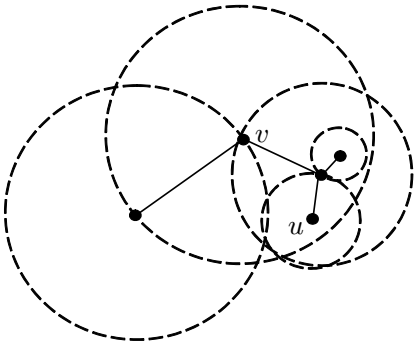

Figure 17.2: A sample topology consisting of five nodes with their corresponding interference radii (dashed circles). Node u experiences interference $I(u) = 2$ since it is covered not only by its direct neighbor but also by node v.

previous chapter. In particular, the interference of a node v is defined as the number of other nodes that potentially affect message reception at node v:

Definition 17.1. *Given a graph $G' = (V, E')$, the interference of a node $v \in V$ is defined as*

$$I(v) = |\{u | u \in V \setminus \{v\}, v \in D(u, r_u)\}|.$$

In other words, the interference I of a node v represents the number of nodes covering v with their disks induced by their transmission ranges set to a value as to reach their farthest neighbors in G'. Note that although each node is also covered by its own disk, we do not consider this kind of self-interference. The node-level interference defined so far is now extended to a graph interference measure as the maximum interference occurring in a graph:

Definition 17.2. *The interference of a graph $G' = (V, E')$ is defined as*

$$I(G') = \max_{v \in V} I(v).$$

Note that Δ, the maximum node degree of the given UDG $G = (V, E)$, is an upper bound for the interference of any subgraph G' of the given graph since in G each node is directly connected to all potentially interfering nodes. However, in arbitrary subgraphs of G the degree of a node only lower-bounds the interference of that node because a node can be covered by non-neighboring nodes (cf. Figure 17.2).

In this chapter we will study the combinatorial optimization problem of finding a resulting topology which maintains connectivity of the given network with minimum interference. Throughout the chapter we will only consider topologies consisting of a tree for each connected component of the given network since additional edges might unnecessarily increase interference.

17.2 Interference in Known Topologies

As justified in the previous section, we restrict our considerations to resulting topologies consisting exclusively of symmetric links (edges). To the best of our knowledge, all currently known topology control algorithms (with the exception of the algorithms presented in Chapter 15) constructing only symmetric connections have in common that every node establishes a link to at least its nearest neighbor. As also mentioned in Chapter 15, this means in a technical sense that these topologies contain the so-called *Nearest Neighbor Forest* as a subgraph. In this section, we will show that this is already a substantial mistake, as thus interference becomes asymptotically incomparable with the interference-minimal topology.

Theorem 17.1. *Any algorithm containing the Nearest Neighbor Forest can have* $\Omega(n)$ *times larger interference than the interference of the optimum connected topology.*

Proof. Our proof uses a node distribution for which the Nearest Neighbor Forest yields interference $\Omega(n)$ while the optimum interference is in $O(1)$. This example has already been studied in Chapter 15, however for a different model.

The Nearest Neighbor Forest for the node distribution depicted in Figure 15.5—assuming that the transmission radius of each node can be chosen sufficiently large—is shown in Figure 15.6. In order to compute the interference, we first observe that an edge from h_i—the ith node in the horizontal node chain—to h_{i+1}—the node directly to its right—covers all nodes to the left, that is all nodes h_j for $j < i$. In particular, the leftmost node h_0 in the horizontal chain is covered by all nodes h_i with $i > 0$. As roughly one third of all nodes are part of the horizontally connected exponential chain, h_0 is covered by at least $\Omega(n)$ nodes.

The optimal tree on the other hand does not connect the horizontal node chain, as depicted in Figure 15.7. The resulting graph has constant interference. □

Although the topology control algorithms presented in Chapter 15 do not necessarily include the Nearest Neighbor Forest, it can be shown that also those algorithms perform badly for our interference model.

17.3 Analysis of the Highway Model

In this section we will study interference for the highway model, in which the node distribution is restricted to one dimension. After analyzing an important artificially constructed problem instance, we will provide a lower bound for interference of general problem instances in the highway model as well as an asymptotically optimal algorithm matching this bound. Finally, an approximation algorithm will be presented.

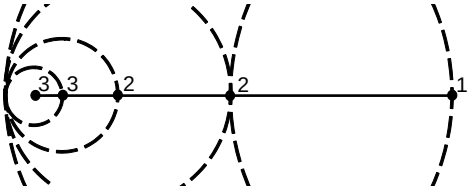

Figure 17.3: Connecting the exponential node chain linearly yields interference $n-2$ at the leftmost node since each node connected to the right covers all nodes to its left. The nodes are labeled according to their experienced interference.

17.3.1 The Exponential Node Chain

How can n nodes arbitrarily distributed in one dimension connect to each other minimizing interference while maintaining connectivity? [74] introduces an instance which seems to yield inherently high interference: The so called *exponential node chain* is a one-dimensional graph $G = (V, E)$ where the distance between two consecutive nodes grows exponentially from left to right as depicted in Figure 15.4. The distance between two nodes v_i and v_{i+1} in V is thus 2^i. Throughout the discussion of the exponential node chain, we furthermore assume that the whole node configuration is normalized in a way that the distance between the leftmost and the rightmost node is not greater than 1: Each node can potentially connect to all other nodes in V and therefore $\Delta = n - 1$, where $n = |V|$. The nodes are termed *linearly connected* if each node—except for the leftmost and the rightmost—maintains an edge to its nearest neighbor to the left and to the right; in other words, node v_i is connected to node v_{i+1} for all $i = 1, \ldots, n-1$ in the resulting topology. In addition to the disks $D(v_i, r_{v_i})$ for each node $v_i \in V$, Figure 17.3 depicts their interference values $I(v_i)$. Since all disks but the one of the rightmost node cover v_1, interference at the leftmost node is $n - 2 \in \Omega(n)$; consequently also interference of the linearly connected exponential node chain is in $\Omega(n)$.

As we will show in the following, the exponential node chain can surprisingly be connected in a significantly better way. According to the construction of the exponential node chain, only nodes connecting to at least one node to their right increase v_1's interference. We call such a node a *hub* and define it as follows:

Definition 17.3. *Given a connected topology for the exponential node chain $G = (V, E)$, a node $v_i \in V$ is defined to be a* hub *in G if and only if there exists an edge (v_i, v_j) with $j > i$.*

The following algorithm $\mathcal{A}_{exp}$ constructs a topology for the exponential node chain G which yields interference $\mathrm{O}(\sqrt{n})$. The algorithm starts with a graph $G_{exp} = (V, E_{exp})$, where V is the set of nodes in the exponential node chain and E_{exp} is initially the empty set. Following the scan-line principle, $\mathcal{A}_{exp}$ processes all nodes in the order of their occurrence from left to right. Initially, the leftmost

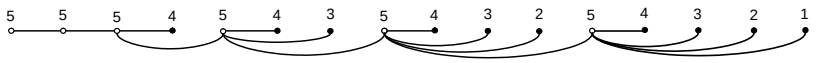

Figure 17.4: The interference of the exponential node chain—shown in a logarithmic scale—is bounded by $O(\sqrt{n})$ by the topology control algorithm $\mathcal{A}_{exp}$. Only hubs (hollow points) interfere with the leftmost node. For clarity of representation, some edges are depicted as curved arcs.

node is set to be the current hub h. Then, for each node v_i, $\mathcal{A}_{exp}$ inserts an edge (h, v_i) into E_{exp}. This is repeated until $I(G_{exp})$ increases due to the addition of such an edge. Now node v_i becomes the current hub and subsequent nodes are connected to v_i as long as the overall interference $I(G_{exp})$ does not increase. Figure 17.4 depicts the resulting topology if $\mathcal{A}_{exp}$ is applied to the exponential node chain. The exponential node chain is thereby depicted in a logarithmic scale.[1] For clarity of representation, some edges in E_{exp} are drawn as curved arcs. In addition, Figure 17.4 shows the individual interference values at each node.

In the following we show that $\mathcal{A}_{exp}$ reduces interference in the exponential node chain.

Theorem 17.2. *Given the exponential node chain G, applying $\mathcal{A}_{exp}$ results in a connected topology with interference $I(G_{exp}) \in O(\sqrt{n})$.*

Proof. The topology resulting to application of $\mathcal{A}_{exp}$ shows a clear structure (cf. Figure 17.4). Each hub, with the exception of the first two, is connected to one more node to its right than its predecessor hub to the left. This follows from the fact that if the current topology leads to interference $I(G_{exp}) = I$ immediately after the determination of a new hub, this hub can be connected to $I - 1$ nodes to its right until $I(G_{exp})$ is again increased by one. Therefore the minimum number of nodes n required in an exponential node chain such that interference $I(G_{exp}) = I$ is obtained, results in

$$n \geq \sum_{i=1}^{I-1} i + 2 = \frac{1}{2}I^2 - \frac{1}{2}I + 2.$$

Solving for I, with $n \geq 2$, we have

$$I \leq \left\lfloor \frac{\sqrt{8n-15}+1}{2} \right\rfloor \in O\left(\sqrt{n}\right).$$

□

This is an intriguing result, since, as we will show in the sequel, $\sqrt{n}$ is a lower bound for the interference of the exponential node chain. In particular, we will

[1] In other words, the exponential node chain is viewed through a pair of glasses with "logarithmic cut".

now prove that there exist network instances where every possible topology exhibits interference at least $\sqrt{n}$. We therefor again consider the exponential node chain introduced in Section 17.3.1 normalized such that all n nodes are located within the line segment of length one.

Theorem 17.3. *Given an exponential node chain $G = (V, E)$ with $n = |V|$, $\sqrt{n}$ is a lower bound for the interference $I(G)$.*

Proof. Let H denote the set of hubs (cf. Definition 17.3) in G and let S be the nodes in $G \setminus H$. In order to prove the theorem, we state two properties for $I(G)$ in the exponential node chain G. First, it holds that $I(G)$ is at least $|H| - 1$, since the leftmost node is interfered with by exactly all hubs except itself (Property 1). On the other hand, $I(G)$ is at least the maximum degree of the resulting topology (Property 2). This holds since a node with maximum degree is covered by at least all disks of its neighboring nodes. We assume for the sake of contradiction that there exists a connected graph that yields interference less than $\sqrt{n}$ for the exponential node chain G. In other words, the degree of any node is required to be at most $\sqrt{n} - 1$, and the number of hubs must not exceed $\sqrt{n}$, including the leftmost node. By the definition of H and S, each node in the graph is either in H or in S, and therefore $|H| + |S| = n$ holds. Due to Property 1, it follows that $|H| \leq \sqrt{n}$. Without loss of generality we assume that the hubs are linearly connected among themselves in order to guarantee connectivity of the graph. Consequently, with Property 2, each hub can connect to at most $\sqrt{n} - 3$ nodes in S (the leftmost and the rightmost hub, respectively, to $\sqrt{n} - 2$). Furthermore, by the definition of a hub, nodes in S are only connected to hubs and not among themselves. Therefore we obtain $|S| \leq \sqrt{n}\,(\sqrt{n} - 3) + 2$. Consequently, $|H| + |S|$ results in $n - 2\sqrt{n} + 2$, which is less than n for $n \geq 2$ and thus leads to a contradiction. □

From Theorems 17.2 and 17.3 it follows that the $\mathcal{A}_{exp}$ algorithm from Section 17.3.1 is asymptotically optimal in terms of interference in the exponential node chain.

17.3.2 The General Highway Model

We considered an important artificially constructed instance in the highway model in the previous section, yielding a lower bound for the interference in arbitrary network graphs. In this section we will go beyond the study of particular network instances and consider arbitrarily distributed nodes in one dimension.

A straightforward question is whether there are instances in the general highway model that are asymptotically worse than the exponential node chain, that is, where a minimum-interference topology exceeds $\Omega(\sqrt{\Delta})$. We answer this question in the negative by introducing the $\mathcal{A}_{gen}$ algorithm, which yields interference in $\mathrm{O}(\sqrt{\Delta})$ for *any* given node distribution.

In a first step, the algorithm determines the maximum degree Δ of the given unit disk graph $G = (V, E)$ and partitions "the highway" into segments of unit

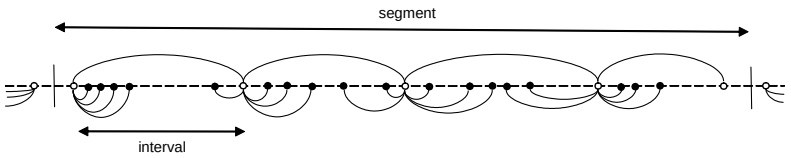

Figure 17.5: $\mathcal{A}_{gen}$ partitions the highway into segments of length 1. In each segment, every $\lceil\sqrt{\Delta}\rceil$-th node becomes a hub (hollow points). While the hubs are connected linearly, each of the remaining nodes in the interval between two hubs is connected to its nearest hub.

length 1. Within such a segment σ, each node can potentially connect to every other node in σ.

In a second step, $\mathcal{A}_{gen}$ considers each segment independently as follows: Starting with the leftmost node of the segment, every $\lceil\sqrt{\Delta}\rceil$-th node (according to their appearance from left to right) becomes a *hub*. A hub is thereby redefined along the lines of Definition 17.3 as a node that has more than one neighboring node, in contrast to *regular* nodes, which are connected to exactly one hub. In order to avoid boundary effects, the rightmost node of each segment is also considered a hub. Then the $\mathcal{A}_{gen}$ algorithm connects the hubs of a segment linearly. That is, each hub, except the leftmost and the rightmost, establishes an edge to its nearest hub to its left and to its right. Two consecutive hubs enclose an *interval*. $\mathcal{A}_{gen}$ connects all regular nodes in a particular interval to their nearest hub—ties are broken arbitrarily. Figure 17.5 depicts one segment of an example instance after the application of $\mathcal{A}_{gen}$. The nodes within a segment form one connected component.

Finally, the $\mathcal{A}_{gen}$ algorithm connects every pair of adjacent segments by connecting the rightmost node of the left segment with the leftmost node of the right segment. This yields a connected topology provided that the corresponding unit disk graph is also connected. Note that with this construction, the hubs may have a comparatively high transmission range (smaller than one unit though). However, the interference range of regular nodes is restricted to their corresponding intervals. This is due to the fact that regular nodes are connected to their nearest hub only, which determines their transmission ranges.

To prove that the resulting topology of $\mathcal{A}_{gen}$ yields $O(\sqrt{\Delta})$ interference, we introduce an additional lemma, which shows that the interference of a node caused by other nodes in the same segment constructed by $\mathcal{A}_{gen}$ is in $O(\sqrt{\Delta})$.

Lemma 17.4. *Each node in a segment σ of the $\mathcal{A}_{gen}$ algorithm experiences at most $O(\sqrt{\Delta})$ interference in the resulting topology of $\mathcal{A}_{gen}$ caused by nodes in σ.*

Proof. By definition of a segment, Δ is an upper bound on the number of nodes in the segment. The $\mathcal{A}_{gen}$ algorithm nominates only every $\lceil\sqrt{\Delta}\rceil$-th node a

hub. Thus, the number of hubs in σ is upper-bounded by $\Delta/\lceil\sqrt{\Delta}\rceil \in \mathrm{O}(\sqrt{\Delta})$. Let hub h_l delimit the interval of a regular node v to the left, and hub h_r to the right, respectively. Furthermore, we can assume without loss of generality that $|h_l v| < |v h_r|$. Therefore $\mathcal{A}_{gen}$ establishes a connection between h_l and v. As this is the only connection of v, it follows that $r_v = |h_l v|$. Consequently, a regular node only interferes with nodes in the same interval. Since a node v is in at most two intervals—hubs are in two intervals—with at most $\lceil\sqrt{\Delta}\rceil$ nodes, v exhibits interference of at most $\mathrm{O}(\sqrt{\Delta})$ regular nodes. Furthermore, v is interfered with by at most $\mathrm{O}(\sqrt{\Delta})$ hubs. □

With this lemma we are ready to prove that the topology constructed by $\mathcal{A}_{gen}$ results in $\mathrm{O}(\sqrt{\Delta})$ interference.

Theorem 17.5. *The resulting topology constructed by the $\mathcal{A}_{gen}$ algorithm from a given graph $G = (V, E)$ yields interference $\mathrm{O}(\sqrt{\Delta})$.*

Proof. By Lemma 17.4, the interference of a detached segment constructed by $\mathcal{A}_{gen}$ is bounded by $\mathrm{O}(\sqrt{\Delta})$. However, interference at node v in segment σ depends also on nodes in the adjacent segments of σ, referred to as σ_l for the segment to the left of σ and σ_r for the segment to the right, respectively. Nodes in other segments do not interfere with v, as the length of a segment is chosen according to the maximum transmission range and thus the interference range of a node is limited to two adjacent segments. We know that at most $\mathrm{O}(\sqrt{\Delta})$ nodes of σ interfere with v. On the other hand, by Lemma 17.4, the rightmost node v' of σ_l is also covered by at most $\mathrm{O}(\sqrt{\Delta})$ disks of nodes in σ_l. This implies that at most $\mathrm{O}(\sqrt{\Delta})$ nodes of σ_l interfere with v since all nodes interfering with v must also cover v' with their disks. By symmetry, the same holds for segment σ_r. Consequently, $\mathcal{A}_{gen}$ results in interference at most three times the interference of an individual segment at each node, which is in $\mathrm{O}(\sqrt{\Delta})$. □

17.3.3 Approximation Algorithm

The $\mathcal{A}_{gen}$ algorithm discussed in the previous section achieves interference in $\mathrm{O}(\sqrt{\Delta})$ for any network instance. This section in contrast introduces an algorithm that approximates the optimum solution for the given network instance. Particularly, it yields interference at most a factor in $\mathrm{O}(\sqrt[4]{\Delta})$ times the interference value resulting from an interference-minimal connectivity-preserving topology.

The $\mathcal{A}_{gen}$ algorithm is in a sense designed for the *worst case*. This is best displayed with an instance where the distances between consecutive nodes are identical. Connecting these nodes linearly, that is connecting each node to its nearest neighbor in each direction, yields constant interference. The $\mathcal{A}_{gen}$ algorithm however constructs a topology resulting in $\mathrm{O}(\sqrt{\Delta})$ interference since a hub connects to one half of the nodes in its corresponding interval for this

instance and an interval contains $\lceil\sqrt{\Delta}\rceil$ nodes. Based on this observation, we introduce the $\mathcal{A}_{apx}$ algorithm, a hybrid algorithm which detects high interference instances and applies $\mathcal{A}_{gen}$ or otherwise connects the nodes linearly.

In the following, we will first present a suitable criterion to identify "high-interference" instances. Given a network graph $G = (V, E)$ in the highway model, let the graph $G_{lin} = (V, E_{lin})$ denote the graph where all nodes in V are linearly connected. For the considered instance to result in high interference at a node v in G_{lin}, many nodes are required to cover v with their corresponding disks. However, with increasing distance to v, these nodes require to have increasing distances to their nearest neighbors in the opposite direction of v in order to interfere with the latter. This leads to an exponential characteristic of these nodes since the edges in E_{lin} accounting for the interference at v form a fragmented exponential node chain. Consequently, the *critical nodes* of v are defined as follows:

Definition 17.4. *Given a linearly connected graph $G_{lin} = (V, E_{lin})$, the critical node set of a node v is defined as*

$$C_v = \{u | u \neq v, |uw| \geq |vu|, \{u, w\} \in E_{lin}\}.$$

In other words, the critical nodes of a node v are those nodes interfering with v if the graph G is connected linearly. Based on the results from Section 17.3.1, we are able to lower-bound the interference of a minimum-interference topology of G as follows.

Lemma 17.6. *Given a graph $G = (V, E)$, let $\gamma = \max_{v \in V} |C_v|$ be the maximum number of critical nodes over all network nodes. A minimum-interference topology for G yields interference in $\Omega(\sqrt{\gamma})$.*

Proof. Let $v \in V$ be the node with maximum interference in G_{lin}. Thus, $|C_v| = \gamma$, as all nodes interfering with v are in C_v. Without loss of generality, we assume that at least half of the nodes in C_v are to the right of v. Let C_v^r be the set of all nodes in C_v to the right of v. We number the nodes $c_i \in C_v^r$ according to their occurrence from left to right. Note that the nodes in C_v^r constitute a *virtual* exponential node chain, as the distance to their nearest neighbor to the right must at least double from c_i to c_{i+1}. Therefore, Theorem 6.1 applies directly to the nodes in C_v^r. Due to the fact that $|C_v^r| \geq |C_v|/2$ and together with Theorem 6.1, we obtain $\Omega(\sqrt{|C_v|})$ as a lower bound for the interference at v. □

The $\mathcal{A}_{apx}$ algorithm makes use of Lemma 17.6 in order to decide whether the existing instance inherently exhibits high interference. In particular, the $\mathcal{A}_{apx}$ algorithm works as follows: $\mathcal{A}_{apx}$ first computes γ. If $\gamma > \sqrt{\Delta}$, $\mathcal{A}_{gen}$ is applied to the graph. Otherwise, if $\gamma \leq \sqrt{\Delta}$, $\mathcal{A}_{apx}$ connects all nodes of the given graph linearly.

Theorem 17.7. *Given a graph G, the $\mathcal{A}_{apx}$ algorithm computes a topology which approximates the optimal interference of G up to a factor in $\mathrm{O}(\sqrt[4]{\Delta})$.*

Proof. We analyze the two possible cases in $\mathcal{A}_{apx}$. In the case $\gamma > \sqrt{\Delta}$, according to Theorem 17.5, $\mathcal{A}_{gen}$ yields interference in $O(\sqrt{\Delta})$. On the other hand, by Lemma 17.6, a minimum-interference topology produces at least $\Omega(\sqrt{\gamma})$ interference. We therefore obtain an approximation ratio in $O(\sqrt{\Delta})/\Omega(\sqrt{\gamma}) \in O(\sqrt[4]{\Delta})$.

In the case $\gamma \leq \sqrt{\Delta}$, by Lemma 17.6, the minimum-interference topology results in interference of at least $\Omega(\sqrt{\gamma})$. Connecting G linearly, we obtain interference γ by definition. Consequently, the approximation ratio of $\mathcal{A}_{apx}$ is in $\gamma/\Omega(\sqrt{\gamma}) \in O(\sqrt[4]{\Delta})$. □

17.4 Concluding Remarks

The results presented in this chapter extend the receiver-centric approach to interference modeling—as studied in the context of sensor networks in Chapter 16—to the analysis of connectivity in general ad hoc networks. The advantages of this interference model are twofold: On the one hand, this definition corresponds to intuition, owing to its receiver-centricity, particularly modeling interference as an effect occurring at the intended receiver of a message, where collisions actually prevent proper reception. On the other hand, this interference model is robust with respect to addition or removal of single nodes, in contrast to the sender-centric interference model proposed in Chapter 15.

Based on this interference model we show that there exist network instances where, to the best of our knowledge, all currently known topology control algorithms (establishing exclusively symmetric connections) fail to effectively confine interference at a low level if required to maintain network connectivity. Led by the observation that already one-dimensional networks exhibit the main complexity of finding low-interference connectivity-preserving topologies, we then focus on the so-called highway model. Starting out to study the special case of the exponential node chain, we finally obtain an algorithm that is guaranteed to always compute a $\sqrt[4]{\Delta}$-approximation of the optimal connectivity-preserving topology in the highway model in general. Adaptation of our approach to higher dimensions remains an open problem and is left for future work.

This receiver-centric approach to the modeling of interference in wireless networks will now be reformulated in a more abstract and at the same time more general way. In particular, the following chapter will formalize interference reduction in cellular networks as a set-based combinatorial optimization problem.

Chapter 18

Interference in Cellular Networks: The Minimum Membership Set Cover Problem

I don't want to belong to any club that will accept me as a member.
Groucho Marx (1890–1977)

This chapter adopts the same receiver-centric approach to interference modeling as the previous two chapters, particularly counting the number of nodes whose transmissions disturb a given network node. It goes beyond those models by accounting for the fact that, typically, transmission ranges—measured in existing wireless networks as the region where the received signal strength lies above a certain threshold—do not resemble perfect circles centered at the sender. On the contrary, effects such as shielding, reflection, and scattering caused by obstacles to signal propagation can lead to observed transmission ranges that barely have anything in common with such an idealistic geometric model. In order to reflect this behavior, transmission ranges will be modeled in this chapter in a general way as sets of affected nodes, considering a sender node and its chosen transmission power level. Based on this model, a combinatorial optimization problem will be formulated to reduce interference in wireless cellular networks.

Strictly speaking, it can be argued that cellular networks—forming the application of the considered problem—lie outside the core scope of this dissertation. We believe however that the presented approach considers the issue of interference from a new perspective and that—interference forming a central issue for wireless networks in general—this approach therefore deserves being discussed in this context.

Cellular networks are heterogeneous networks consisting of two different types of nodes: base stations and clients. The base stations—acting as servers—are interconnected by an external fixed backbone network; clients are connected via radio links to base stations. The totality of the base stations forms the infrastructure for distributed applications running on the clients, the most prominent of which probably being mobile telephony. Cellular networks can however more broadly be considered a type of infrastructure for distributed tasks in general.

Since communication over the wireless links takes place in a shared medium, interference can occur at a client if it is within transmission range of more than one base station. In order to prevent such collisions, coordination among the conflicting base stations is required. Commonly, this problem is solved by segmenting the available frequency spectrum into channels to be assigned to the base stations in such a way as to prevent interference, in particular such that no two base stations with overlapping transmission range use the same channel.

In this chapter we will assume a different approach to interference reduction. The basis of our analysis is formed by the observation that interference effects occurring at a client depend on the number of base stations by whose transmission ranges it is covered. In particular for solutions using frequency division multiplexing as described above, the number of base stations covering a client is a lower bound for the number of channels required to avoid conflicts; a reduction in the required number of channels, in turn, can be exploited to broaden the frequency segments and consequently to increase communication bandwidth. On the other hand, also with systems using code division multiplexing, the coding overhead can be reduced if only a small number of base stations cover a client.

The transmission range of a base station—and consequently the coverage properties of the clients—depends on its position, obstacles hindering the propagation of electromagnetic waves, such as walls, buildings, or mountains, and the base station transmission power. Since due to legal or architectural constraints the former two factors are generally difficult to control, we assume a scenario in which the base station positions are fixed, where each base station can however adjust its transmission power. The problem of minimizing interference then consists in assigning a transmission power level to every base station such that the number of base stations covering any node is minimal (cf. Figure 18.1). At the same time, it has to be guaranteed that every client is covered by at least one base station in order to maintain availability of the network.

In Figure 18.1 the area covered by a base station b transmitting with a given power level is represented by a disk centered at b and having a radius corresponding to the chosen transmission power. Practical measurements however show that this idealization is far from realistic. Not only mechanical and electronic inaccuracies inevitable in the construction of antennas, but more importantly the presence of obstacles to the propagation of electromagnetic signals—such as buildings, mountains, or even weather conditions—can lead to areas covered by signal transmission that hardly resemble disks in practice. These considerations

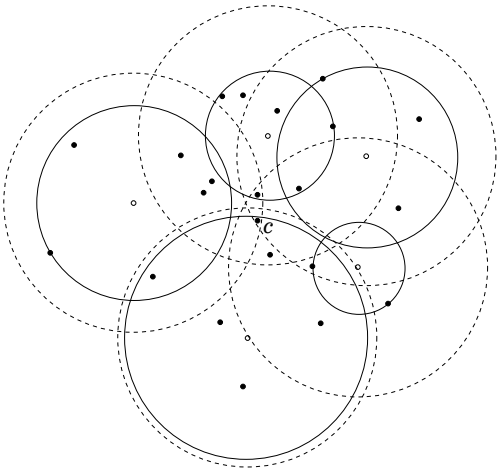

Figure 18.1: If the base stations (hollow points) are assigned identical transmission power levels (dashed circles), client c experiences high interference since it is covered by all base stations. Interference can be reduced by assigning appropriate power values (solid circles) such that all clients are covered by at most two base stations.

suggest that, in order to study the described interference reduction problem, we abstract from network node positions and circular transmission areas.

In our analysis, we will formalize the task of reducing interference as a combinatorial optimization problem. For this purpose we will model the transmission range of a base station having chosen a specific transmission power level as a set containing exactly all clients covered thereby. The totality of transmission ranges selectable by all base stations is consequently modeled as a collection of client sets. More formally, this yields the *Minimum Membership Set Cover (MMSC)* problem: Given a set of elements U (modeling clients) and a collection S of subsets of U (transmission ranges), choose a solution $S' \subseteq S$ such that every element occurs in at least one set in S' (maintain network availability) and that the *membership* $M(e, S')$ of any element e with respect to S' is minimal, where $M(e, S')$ is defined as the number of sets in S' in which e occurs (interference).

Having defined this formalization, we will show in this chapter—by reduction from the related Minimum Set Cover problem—that the MMSC problem is *NP*-complete and that no polynomial-time algorithm exists with approximation ratio less than $\ln n$ unless $NP \subset TIME(n^{O(\log \log n)})$.[1] We additionally present

[1] $TIME(t)$ denotes the set of languages that have a deterministic algorithm running in time t.

a probabilistic algorithm based on linear programming relaxation asymptotically matching this lower bound, particularly yielding an approximation ratio in $O(\log n)$ with high probability. Furthermore we study how the presented algorithm performs in practical network instances.

The remainder of the chapter is organized as follows: The MMSC problem being formally defined in Section 18.1, the subsequent section contains a description of the lower bound with respect to approximability of the MMSC problem. In the following section we will describe how the MMSC problem can be formulated as a linear program and will provide a $O(\log n)$-approximation algorithm for the problem. The behavior of the proposed algorithm in practical networks is the subject of Section 18.4.

18.1 Minimum Membership Set Cover

As described in the introduction to this chapter, the problem considered in this chapter is to assign to each base station a transmission power level such that interference is minimized while all clients are covered. For our analysis we formalize this problem by introducing a combinatorial optimization problem referred to as *Minimum Membership Set Cover*. In particular, clients are modeled as elements and the transmission range of a base station given a certain power level is represented as the set of thereby covered elements. In the following, we will first define the membership of an element given a collection of sets:

Definition 18.1 (Membership). *Let U be a finite set of elements and S be a collection of subsets of U. Then the membership $M(u, S)$ of an element u is defined as $|\{T \mid u \in T, T \in S\}|$.*

Informally speaking, MMSC is identical to the MSC problem apart from the minimization function. Where MSC minimizes the total number of sets, MMSC tries to minimize element membership. Particularly, MMSC can be defined as follows:

Definition 18.2 (Minimum Membership Set Cover). *Let U be a finite set of elements with $|U| = n$. Furthermore let $S = \{S_1, \ldots, S_m\}$ be a collection of subsets of U such that $\bigcup_{i=1}^{m} S_i = U$. Then* Minimum Membership Set Cover *(MMSC) is the problem of covering all elements in U with a subset $S' \subseteq S$ such that $\max_{u \in U} M(u, S')$ is minimal.*[2]

Note that—as discussed in the introduction of the chapter—the problem statement does not require the collection of subsets S to reflect geometric posi-

[2]Besides minimizing the *maximal* membership value over all elements, also minimization of the *average* membership value can be considered a reasonable characterization of the interference reduction problem. The fact however that—given a solution S'—the sum of all membership values equals the sum of the cardinalities of the sets in S' shows that this min-average variant is identical to the Weighted Set Cover [22] problem with the set weights corresponding to set cardinalities.

tions of network nodes. For a given problem instance to be valid, $\bigcup_{i=1}^{m} S_i = U$ is sufficient.

18.2 Problem Complexity

In this section we will address the complexity of the *Minimum Membership Set Cover* problem. We will show that MMSC is *NP*-complete and therefore no polynomial time algorithm exists that solves MMSC unless $P = NP$.

Theorem 18.1. *MMSC is NP-complete.*

Proof. We will prove that MMSC is *NP*-complete by reducing MSC to MMSC. Consider an MSC instance (U, S) consisting of a finite set of elements U and a collection S of subsets of U. The objective is to choose a subset S' with minimum cardinality from S such that the union of the chosen subsets of U contains all elements in U.

We now define a set $\widetilde{U}$ by adding a new element u to U, construct a new collection of sets $\widetilde{S}$ by inserting u into all sets in S, and consider $(\widetilde{U}, \widetilde{S})$ as an instance of MMSC. Since the element u is in every set in $\widetilde{S}$, it follows that u is an element with maximum membership in the solution S' of MMSC. Moreover, the membership of u in S' is equal to the number of sets in the solution. Therefore MMSC minimizes the number of sets in the solution by minimizing the membership of u. Consequently we obtain the solution for MSC of the instance (U, S) by solving MMSC for the instance $(\widetilde{U}, \widetilde{S})$ and extracting the element u from all sets in the solution.

We have shown a reduction from MSC to MMSC, and therefore the latter is *NP*-hard. Since solutions for the decision problem of MMSC are verifiable in polynomial time, it is in *NP*, and consequently the MMSC decision problem is also *NP*-complete. □

Now that we have proved MMSC to be *NP*-complete and therefore not to be optimally computable within polynomial time unless $P = NP$, the question arises, how closely MMSC can be approximated by a polynomial time algorithm. This is partly answered with the following lower bound.

Theorem 18.2. *There exists no polynomial time approximation algorithm for the MMSC problem with an approximation ratio less than* $\ln n$ *unless* $NP \subset TIME(n^{\mathrm{O}(\log \log n)})$.

Proof. The reduction from MSC to MMSC in the proof of Theorem 18.1 is approximation-preserving, that is, it implies that any lower bound for MSC also holds for MMSC. In [34] it is shown that $\ln n$ is a lower bound for the approximation ratio of MSC unless $NP \subset TIME(n^{\mathrm{O}(\log \log n)})$. Thus, $\ln n$ is also a lower bound for the approximation ratio of MMSC. □

18.3 Approximating MMSC by LP Relaxation

In the previous section, a lower bound of $\ln n$ for the approximability of the MMSC problem by means of polynomial time approximation algorithms has been established. In this section we will show how to obtain a $O(\log n)$-approximation with high probability[3] using LP relaxation techniques. For an introduction to linear programming see for instance [23].

18.3.1 LP Formulation of MMSC

We will first derive the integer linear program which describes the MMSC problem and then formulate the linear program that relaxes the integrality constraints.

Let $S' \subseteq S$ denote a subset of the collection S. To each $S_i \in S$ we assign a variable $x_i \in \{0,1\}$ such that $x_i = 1 \Leftrightarrow S_i \in S'$. For S' to be a set cover, it is required that for each element $u_i \in U$ at least one set S_j with $u_i \in S_j$ is in S'. Therefore, S' is a set cover of U if and only if for all $i = 1, \ldots, n$ it holds that $\sum_{S_j : u_i \in S_j} x_j \geq 1$. For S' to be minimal in the number of sets that cover a particular element, we need a second set of constraints. Let z be the maximum membership over all elements caused by the sets in S'. Then for all $i = 1, \ldots, n$ it follows that $\sum_{S_j : u_i \in S_j} x_j \leq z$. The MMSC problem can consequently be formulated as the integer program $\mathrm{IP}_{\mathrm{MMSC}}$:

$$
\begin{aligned}
&\text{minimize } z \\
&\text{subject to} \quad \sum_{S_j : u_i \in S_j} x_j \geq 1 \qquad i = 1, ..., n \\
&\qquad\qquad \sum_{S_j : u_i \in S_j} x_j \leq z \qquad i = 1, ..., n \\
&\qquad\qquad x_j \in \{0,1\} \qquad j = 1, ..., m
\end{aligned}
$$

By relaxing the constraints $x_j \in \{0,1\}$ to $x'_j \geq 0$, we obtain the following linear program $\mathrm{LP}_{\mathrm{MMSC}}$:

[3] Throughout this chapter, an event E occurring "with high probability" stands for $Pr[E] = 1 - O(\frac{1}{n})$.

$$\begin{aligned} &\text{minimize } z \\ &\text{subject to} \quad \sum_{S_j : u_i \in S_j} x'_j \geq 1 \qquad i = 1, ..., n \\ &\qquad\qquad \sum_{S_j : u_i \in S_j} x'_j \leq z \qquad i = 1, ..., n \\ &\qquad\qquad x'_j \geq 0 \qquad j = 1, ..., m \end{aligned}$$

The integer program IP_{MMSC} yields the optimal solution z^* for an MMSC problem. The derived linear program LP_{MMSC} therefore obtains a fractional solution z' with $z' \leq z^*$ since we allow the variables x'_j to be in [0,1].

18.3.2 Algorithm and Analysis

We will now present a $O(\log n)$-approximation algorithm, referred to as $\mathcal{A}_{\text{MMSC}}$, for the MMSC problem. Given an MMSC instance (U, S), the algorithm first solves the linear program LP_{MMSC} corresponding to (U, S). In a second step, $\mathcal{A}_{\text{MMSC}}$ performs randomized rounding (see [93]) on a feasible solution vector $\underline{x}'$ for LP_{MMSC} in order to derive a vector $\underline{x}$ with $x_i \in \{0, 1\}$. Finally it is ensured that $\underline{x}$ is a feasible solution for IP_{MMSC} and consequently a set cover.

Algorithm $\mathcal{A}_{\text{MMSC}}$

Input: an MMSC instance (U, S)

1: compute solution vector $\underline{x}'$ to the linear program LP_{MMSC} corresponding to (U, S)
2: $p_i := \min\{1,\ x'_i \cdot \log n\}$
3: $x_i := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
4: **for all** $u_i \in U$ **do**
5: **if** $\sum_{S_j : u_i \in S_j} x_j = 0$ **then**
6: set $x_j = 1$ for any j such that $u_i \in S_j$
7: **end if**
8: **end for**

Output: MMSC solution S' corresponding to $\underline{x}$

For the analysis of $\mathcal{A}_{\text{MMSC}}$ the following two mathematical facts are required. Their proofs are omitted and can be found in mathematical text books.

Fact 18.1. (Means Inequality) *Let $\mathcal{A} \subset \mathbb{R}^+$ be a set of positive real numbers. The product of the values in $\mathcal{A}$ can be upper-bounded by replacing each factor*

with the arithmetic mean of the elements of $\mathcal{A}$:

$$\prod_{x \in \mathcal{A}} x \;\leq\; \left(\frac{\sum_{x \in \mathcal{A}} x}{|\mathcal{A}|} \right)^{|\mathcal{A}|}.$$

Fact 18.2. *For all n, t, such that $n \geq 1$ and $|t| \leq n$,*

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \;\leq\; e^t.$$

We will prove $\mathcal{A}_{\text{MMSC}}$ to be a $O(\log n)$-approximation algorithm for IP_{MMSC} in several steps. We first show that the membership of an element in U after the randomized rounding step of $\mathcal{A}_{\text{MMSC}}$ is bounded with high probability.

Lemma 18.3. *The membership of an element u_i after Line 3 of $\mathcal{A}_{\text{MMSC}}$ is at most $2e \log n \cdot z^*$ with high probability.*

Proof. The optimal solution of LP_{MMSC} leads to fractional values x'_j and does not admit a straightforward choice of the sets S_j. Using randomized rounding, $\mathcal{A}_{\text{MMSC}}$ converts the fractional solution to an integral solution S'. In Line 3, a set S_j is chosen to be in S' with probability $x'_j \cdot \log n$. Thus, the expected membership of an element u_i is

$$E[M(u_i, S')] = \sum_{S_j : u_i \in S_j} x'_j \cdot \log n \leq \log n \cdot z'. \tag{18.1}$$

The last inequality follows directly from the second set of constraints of LP_{MMSC}. Since $z' \leq z^*$, it follows that the expected membership for u_i is at most $\log n \cdot z^*$. Now we need to ensure that, with high probability, u_i is not covered too often. Since randomized rounding can be modeled as Poisson trials, we are able to use a Chernoff bound [78]. Let Y_i be a random variable denoting the membership of u_i with expected value $\mu = E[M(u_i, S')]$. Applying the Chernoff bound, we derive

$$Pr\left[Y_i \geq (1+\delta)\,\mu\right] < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\mu.$$

Choosing $\delta \geq 2e - 1$, the right hand side of the inequality simplifies to

$$\left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\mu \leq \left(\frac{e^\delta}{(2e)^{(1+\delta)}} \right)^\mu < \left(\frac{e^\delta}{(2e)^\delta} \right)^\mu = 2^{-\delta\mu}. \tag{18.2}$$

Since the above Chernoff bound corresponds to the upper tail of the probability distribution of Y_i and as μ is at most $\log n \cdot z^*$, it follows that

$$Pr\left[Y_i \geq (1+\delta) \log n \cdot z^*\right] \leq Pr\left[Y_i \geq (1+\delta)\,\mu\right].$$

However, for this inequality to hold, only $(1+\delta)\mu \leq c \log n \cdot z^*$ for some constant c is required. Thus, by setting $(1+\delta)\mu = c \log n \cdot z^*$ and using Inequality (18.1), we obtain

$$\delta\mu \quad \geq \quad (c-1)\log n \cdot z^*. \tag{18.3}$$

Using Inequalities (18.2) and (18.3), we can then bound the probability that the membership of u_i is greater than $c \log n \cdot z^*$ as follows:

$$Pr\left[Y_i \geq c \log n \cdot z^*\right] < 2^{-\delta\mu} \leq 2^{-(c-1)\log n \cdot z^*} = \frac{1}{n^{(c-1)z^*}}.$$

In order to compute c, we again consider the equation $(1+\delta)\mu = c \log n \cdot z^*$. Solving for δ, we derive

$$\delta = \frac{c \log n \cdot z^*}{\mu} - 1.$$

As a requirement for Inequality (18.2) we demand δ to be greater or equal to $2e-1$. Furthermore, the right hand side of the inequality is minimal if μ is maximal. Thus, using Inequality (18.1), we obtain

$$\frac{c \log n \cdot z^*}{\log n \cdot z^*} - 1 \quad \geq \quad 2e-1$$

or $c \geq 2e$. Taking everything together and using $z^* \geq 1$, it follows that

$$Pr\left[Y_i \geq 2e \log n \cdot z^*\right] < \frac{1}{n^{(2e-1)z^*}} \in \mathrm{O}\left(\frac{1}{n^4}\right).$$

□

Now we are ready to show that after randomized rounding all elements have membership at most $2e \log n \cdot z^*$ with high probability.

Lemma 18.4. *The membership of all elements in U after Line 3 of $\mathcal{A}_{\mathrm{MMSC}}$ is at most $2e \log n \cdot z^*$ with high probability.*

Proof. Let E_i be the event that the membership of element u_i after Line 3 of $\mathcal{A}_{\mathrm{MMSC}}$ is greater than $2e \log n \cdot z^*$. Then the probability that the membership for all elements in U is less than $2e \log n \cdot z^*$ equals

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,].$$

We know from Lemma 18.3 that the probability $\Pr[E_i]$ is less than $1/n^{(2e-1)z^*}$. Since the events are clearly not independent, we cannot apply the product rule. However, it was shown in [100] that

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] \geq \prod_{i=1}^{n} Pr[\,\overline{E_i}\,]. \tag{18.4}$$

We can make use of this bound, as $\mathrm{IP}_{\mathrm{MMSC}}$ features the *positive correlation* property assumed in [100]. Consequently, setting $\alpha = (2e-1)z^*$ and using Inequality (18.4), it follows that

$$\begin{aligned} Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] &\geq \left(1-\frac{1}{n^\alpha}\right)^n = \left(1-\frac{1}{n^\alpha}\right)^{\frac{n^\alpha-1}{n^{\alpha-1}-\frac{1}{n}}} \\ &\geq e^{-\frac{1}{n^{\alpha-1}-\frac{1}{n}}} > 1-\frac{1}{n^{\alpha-1}-\frac{1}{n}}. \end{aligned}$$

For the third inequality we use Fact 18.2 with $t=-1$, which leads to the inequality

$$e^{-1} \leq (1-1/n)^{n-1}.$$

The last inequality is derived by Taylor series expansion of the left hand term. Consequently, using $\alpha = (2e-1)z^*$ and $z^* \geq 1$, we obtain

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] = 1 - \mathrm{O}\left(\frac{1}{n^3}\right).$$

□

Since $\mathcal{A}_{\mathrm{MMSC}}$ uses randomized rounding, we do not always derive a feasible solution for $\mathrm{IP}_{\mathrm{MMSC}}$ after Line 3 of the algorithm. That is, there exist elements in U that are not covered by a set in S'. But we can show in the following lemma that each single element is covered with high probability.

Lemma 18.5. *After Line 3 of $\mathcal{A}_{\mathrm{MMSC}}$, an element u_i in U is covered with high probability.*

Proof. For convenience we define C_i to be the set $\{S_j \mid u_i \in S_j\}$. From $\mathrm{LP}_{\mathrm{MMSC}}$ we know that $\sum_{S_j \in C_i} x'_j \geq 1$. Thus, it follows that

$$\sum_{S_j \in C_i} p_j \geq \log n. \tag{18.5}$$

Let q_i be the probability that an element u_i is contained in none of the sets in S' obtained by randomized rounding, that is, $q_i = Pr\,[M(u_i, S') = 0]$. Consequently we have

$$\begin{aligned} q_i &= \prod_{S_j \in C_i} (1-p_j) \leq \left(1-\frac{\sum_{S_j \in C_i} p_j}{|C_i|}\right)^{|C_i|} \\ &\leq e^{-\sum_{S_j \in C_i} p_j} \leq e^{-\log n} = \frac{1}{n}. \end{aligned}$$

The first inequality follows from Fact 18.1, the second inequality is a consequence of Fact 18.2, and the third step is derived from Inequality (18.5). □

Lines 4 to 8 of $\mathcal{A}_{\text{MMSC}}$ ensure that the final solution S' is a set cover. This is achieved by consecutively including sets in S' until all elements are covered. In the following we show that the additional maximum membership increase caused thereby is bounded with high probability.

Lemma 18.6. *In Lines 4 to 8 of $\mathcal{A}_{\text{MMSC}}$, the maximum membership in U is increased by at most $\mathrm{O}(\log n)$ with high probability.*

Proof. In order to bound the number of sets added in the considered part of the algorithm, again a Chernoff bound is employed. Let Z be a random variable denoting the number of uncovered elements after Line 3 of $\mathcal{A}_{\text{MMSC}}$. From Lemma 18.5 we know that an element is uncovered after randomized rounding with probability less than $1/n$. Consequently, the expected value μ for Z is less than 1. Using a similar analysis as in Lemma 18.3, we obtain

$$Pr\left[Z \geq c\right] < 2^{-c+1},$$

where $c \geq 2e$ is required. Setting $c = \log n + 2e$, it follows that

$$Pr\left[Z \geq \log n + 2e\right] \;<\; \frac{2}{n \cdot 4^e} \;\in\; \mathrm{O}\left(\frac{1}{n}\right).$$

The proof is concluded by the observation that each additional set added in the second step of $\mathcal{A}_{\text{MMSC}}$ increases the maximum membership in U by at most one. Since only $\mathrm{O}(\log n)$ elements have to be covered with high probability and as it is sufficient to add one set per element, the lemma follows.[4] □

Now we are ready to prove that $\mathcal{A}_{\text{MMSC}}$ yields a $\mathrm{O}(\log n)$-approximation for IP_{MMSC} and consequently also for MMSC.

Theorem 18.7. *Given an MMSC instance consisting of m sets and n elements, $\mathcal{A}_{\text{MMSC}}$ computes a $\mathrm{O}(\log n)$-approximation with high probability. The running time of $\mathcal{A}_{\text{MMSC}}$ is polynomial in $m \cdot n$.*

Proof. The approximation factor in the theorem directly follows from Lemmas 18.4 and 18.6. The running time result is a consequence of the existence of algorithms solving linear programs in time polynomial in the program size [56] and to the fact that LP_{MMSC} can be described using -1, 0, and 1 as coefficients only. □

18.3.3 Alternative Algorithm

In an alternative version of the algorithm, the values $\underline{x}'$ obtained by solving LP_{MMSC} can be directly employed as probabilities for randomized rounding

[4] Since in the above Chernoff bound μ is at most a constant, a more careful analysis would yield that the maximum membership in U is increased—with high probability—by $\mathrm{O}(\log n / \log\log n)$ only. This improvement has however no impact on the main result of this chapter.

(without the additional factor of $\log n$). In this case, randomized rounding is repeated for all sets containing elements not yet covered until resulting in a set cover. With similar arguments as for $\mathcal{A}_{\text{MMSC}}$, it can be shown that this modified algorithm achieves the same approximation factor and that it terminates after repeating randomized rounding at most $\log n$ times, both with high probability.

18.4 Average-Case Networks

While the previous section showed that $\mathcal{A}_{\text{MMSC}}$ approximates the optimal solution up to a factor in $O(\log n)$, this section will discuss average-case networks. In particular, the algorithms $\mathcal{A}_{\text{MMSC}}$ and $\widetilde{\mathcal{A}}_{\text{MMSC}}$—the alternative algorithm described in Section 18.3.3—are considered. Since the approximation performance of algorithms is studied, we will denote by the *membership of a solution* the minimization function value—that is the maximum membership over all clients—of the corresponding MMSC solution.

The studied algorithms were executed on instances generated by placing base stations and clients randomly according to a uniform distribution on a square field with side length 5 units. Adaptable transmission power values were modeled by attributing to each base station circles with radii 0.25, 0.5, 0.75, and 1 unit; each such circle then contributes one set containing all covered clients to the problem instance thereafter presented to the algorithms. The fact that the $\mathcal{A}_{\text{MMSC}}$ and $\widetilde{\mathcal{A}}_{\text{MMSC}}$ algorithms can choose more than one circle attributed to the same base station would imply the necessity of an additional post-processing step to eliminate these cases. Since this however appears to occur rarely given the generated instances, such post-processing has been omitted in our implementations.

As shown in the previous section, the approximation factor of the algorithms depends on the number of clients. For this reason the simulations were carried out over a range of client densities. Since the membership value obtained by solving LP_{MMSC} lies below the optimal solution and therefore the gap between the algorithm result and the solution of the linear program is an upper bound for the obtained approximation ratio, the LP_{MMSC} result z' is also considered.

For a base station density of 2 base stations per unit disk, Figure 18.2(a) shows the mean membership values over 200 networks—for each simulated client density—for the results computed by $\mathcal{A}_{\text{MMSC}}$, $\widetilde{\mathcal{A}}_{\text{MMSC}}$, and the values obtained by solving LP_{MMSC}. The results depict that for this relatively low base station density all measured values are comparable and increase with growing client density. In contrast, for a higher base station density of 5 base stations per unit disk (cf. Figure 18.2(b)), a gap opens between the $\mathcal{A}_{\text{MMSC}}$ and LP_{MMSC} results. Whereas the ratio between these two result series—as mentioned before, an upper bound for the approximation ratio—rises sharply for low client densities, its increase diminishes for higher client densities, which corresponds to the $O(\log n)$ approximation factor described in the theoretical analysis. Additionally, it can be observed that $\widetilde{\mathcal{A}}_{\text{MMSC}}$ performs significantly better than $\mathcal{A}_{\text{MMSC}}$. The reason

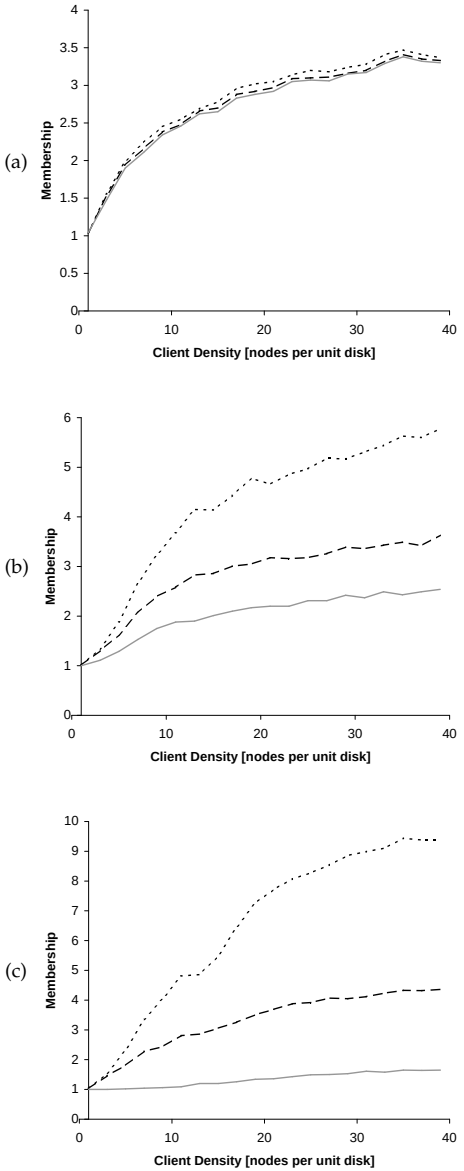

Figure 18.2: Mean values of the membership results obtained by $\mathcal{A}_{\mathrm{MMSC}}$ (dotted), $\widetilde{\mathcal{A}}_{\mathrm{MMSC}}$ (dashed), and the $\mathrm{LP}_{\mathrm{MMSC}}$ solution with 2 (a), 5 (b), and 10 (c) base stations per unit disk.

for this effect lies in the fact that $\mathcal{A}_{\mathrm{MMSC}}$ multiplies the $\underline{x}'$ values resulting from $\mathrm{LP}_{\mathrm{MMSC}}$ by the factor $\log n$ to obtain the probabilities employed for randomized rounding, whereas this multiplication is not performed by $\widetilde{\mathcal{A}}_{\mathrm{MMSC}}$. The approximation gap becomes even wider for higher base station densities, such as 10 base stations per unit disk (Figure 18.2(c)). Our simulations showed however that beyond this base station density no significant changes in the membership results can be observed.

The increasing gap between the simulated algorithms and the $\mathrm{LP}_{\mathrm{MMSC}}$ solution with growing base station density can be explained by the following observation: For low base station densities—where problem instances contain a small number of sets—a relatively large number of clients are covered by only one set, which consequently will have to be chosen in both the $\mathrm{LP}_{\mathrm{MMSC}}$ and the algorithm solutions; for high base station densities, in contrast, the solution weights $\underline{x}'$ computed by $\mathrm{LP}_{\mathrm{MMSC}}$ can be distributed more evenly among the relatively high number of available sets, and the potential of "committing an error" during randomized rounding increases.

In summary, the simulations show that the considered algorithms approximate the optimum solution well in average-case networks. As regards comparison of $\mathcal{A}_{\mathrm{MMSC}}$ and $\widetilde{\mathcal{A}}_{\mathrm{MMSC}}$, it can be observed that, in practice, the latter algorithm performs even better than the former.

18.5 Concluding Remarks

Interference reduction in cellular networks is studied in this chapter by means of formalization with the *Minimum Membership Set Cover* problem. To the best of our knowledge, this combinatorial optimization problem has not been studied before. In particular, we show in this chapter, using an approximation-preserving reduction from the Minimum Set Cover problem, that MMSC is not only NP-hard, but also that no polynomial-time algorithm can approximate the optimal solution more closely than up to a factor $\ln n$ unless $NP \subset TIME(n^{\mathrm{O}(\log \log n)})$. In a second part of the chapter, this lower bound is shown to be asymptotically matched by a randomized algorithm making use of linear programming relaxation techniques. The third part of the chapter discusses the behavior of the algorithm in average-case networks. In particular, it shows that the algorithm can be modified to perform well not only in the worst case but also in the average case. Finally, the question remains as an open problem whether there exists a simpler greedy algorithm—considering interference increase during its execution—with the same approximation quality.

Although the application of the problem discussed in this chapter goes beyond the scope of this dissertation in a strict sense, this approach to interference reduction in wireless networks has been presented owing to its character differing from the techniques presented in the previous chapters. Besides this approach, we have studied in this second part of the dissertation different models and

problem formulations for the task of reducing interference in various types of wireless networks. Although the different models of interference are all justified where they are introduced, it is true, at least certain definition details may sometimes appear arbitrary. Indeed,—even if restricting oneself to model definitions independent of network traffic and where message transmission does or does not interfere with other nodes in a simplistic binary manner—many more models for interference exist, some more natural, others less reasonable. The following chapter will present and discuss a systematic overview of different interference models in the context of topology control based on transmission power control.

Chapter 19

On Modeling Interference

The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work.
John von Neumann (1903–1957)

The core issue considered in Chapters 15 to 18 is interference. In contrast to most of the previous work, we thereby study interference as an explicit network property. Such explicitness, in turn, requires exact definition of what is understood by interference. We have restricted our studies to interference models with the following idealizations:

- Interference is binary in the sense that for every transmitted signal a subset of the network nodes (or edges, as explained later) is interfered with, whereas the remaining nodes are not affected.

- Interference is independent of network traffic. Instead it is based on chosen transmission ranges.

- Transmission ranges reflect selected connections in the network.

The first assumption is clearly a stark simplification of reality; nevertheless, as the previous chapters show, such a binary model can allow for intuitive results and can be considered a first step towards understanding interference. Independence of network traffic, as stated in the second assumption, is desirable, since network traffic highly depends on the chosen application and as its characteristics are commonly not known prior to network operation. The third assumption does not imply that transmission ranges have a certain geometric shape. In Chapters 15, 16, and 17, transmission ranges correspond to chosen

edges in the resulting network graph; in Chapter 18, transmission ranges reflect potential links between mobile nodes and base stations.

Based on these model assumptions, this chapter aims at providing an overview and comparison of various interference models, some of which are more reasonable, others less intuitive, some allow for problems which are optimally solvable in polynomial time, others are inherently difficult. While the most intuitive and best justifiable interference models have been adopted for analysis in the previous chapters, this chapter is intended to provide an overview from a "model architect's" perspective.

As mentioned earlier, interference reduction per se is meaningless, as the trivial solution of having all nodes send with least possible power (or have them shut down their radio devices) minimizes interference. The problem of interference minimization only becomes reasonable if certain properties are stated as requirements to be fulfilled by the resulting topologies. As shown in the previous chapters, typical requirements to be met by resulting topologies are network connectivity, the spanner property, or planarity, while, in principle, any network property is possible to be stipulated. In this chapter, we will consider interference independent of any required network property. In other words, interference is regarded as a measure given a resulting network topology with specified connections—or more exactly a graph with specified edges.

One of the first questions that should be asked when modeling interference is who interferes with whom. Is it mainly nodes interfering with other nodes or do links disturb other links? Is there a significant difference between the two perspectives? At first sight it may seem clear that nodes—sending messages—are the sources for interference. On the other hand it can also be argued that communication along a link imposes interference to all nodes within the vicinity of the link and hence interference from edges onto nodes should be considered. But then again, what is prevented by interference is communication over links. Partitioned into four sections corresponding to the categories *node-to-node*, *edge-to-edge*, *edge-to-node*, and *node-to-edge* interference, several more questions will be discussed. Where shall interference be measured, at the originators or rather at the affected nodes? Does it make a difference at all? How do we derive a measure for the entire network from local measures?

19.1 Node-to-Node Interference

Network nodes are the physical entities that eventually emit radio signals and are disturbed by signals of other stations. It can therefore be argued that interference should best be defined to occur between nodes. Focusing on nodes as causing interference, this section discusses directed links—or directed edges in terms of graphs—, as a transmitted message is considered to be independent of a possible reply.

Figure 19.1 shows an overview of several additional decisions that are required along the way to a model of interference. The displayed interference

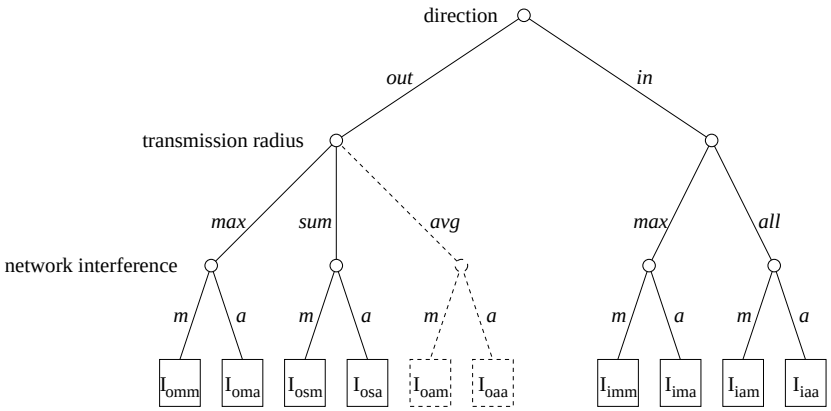

Figure 19.1: Node-to-node interference.

models are defined according to the three parameters *direction*, *transmission radius*, and *network interference*. Choosing the *out* branch at the *direction* node corresponds to measuring interference at the originators counting the affected nodes (*outgoing interference*), whereas the *in* branch corresponds to measuring interference at the affected nodes counting the nodes by which a considered node is affected (*incoming interference*).

The next decision parameter—*transmission radius*—is concerned with the way interference at a node is defined. We will first look at the *out* subtree, that is, for each node, the impact its activity has onto other nodes is considered. A message sent along an edge e interferes with all nodes lying within the transmission range corresponding to the edge e, or in an idealized way, with all nodes contained in the disk of radius $|e|$ centered at the sending node. But how can the interference of a node with several incident edges be defined? Figure 19.1 provides three possibilities of assigning a value to a node v:

max A disk with its radius corresponding to the longest incident edge of v is chosen. The number of nodes covered by this disk is the value for v.

sum For all incident edges of v, the quantities of nodes covered by the corresponding disks are summed up.

avg Similar to the *sum* case, but the value is finally divided by the number of edges incident to v.

Regardless of the definition of the interference measure of a node, when stepping onwards to the entire network graph, some pooling of the individual node interference values is required. This can for instance be done by taking the maximum (m) or the average (a) of all nodes, as shown in the figure. The first option tries to reflect interference in worst-case regions of the network, whereas

the second aims at modeling how much the network as a whole is exposed to interference. Alternatively, the sum over all nodes could be computed, which would however lead to a counter-intuitive model having a direct dependency on the number of nodes in the graph.

For the *in* subtree, the *transmission radius* parameter offers two options:

max As in the *max* case of the *out* subtree, a disk of maximum radius (over all incident edges) is considered for each node in the graph. The value assigned to a node v is now the number of disks that cover v.

all Not only the longest edge incident to a node is considered, but every edge e is represented by a disk with its radius equal to $|e|$. Again the number of disks covering v is counted.

At the bottom of the decision tree we also have the maximum (m) and average (a) pooling options.

Node-to-node interference models based on incoming interference are studied in Chapters 16 and 17, particularly I_{imm} (defined as *in-max-m*) in sensor networks and general ad hoc networks, respectively. As Chapter 16 considers minimum-interference sink trees, where every node has exactly one outgoing directed edge, the I_{iam} or also an edge-to-node interference model as discussed below would lead to the same interference values. Although the problem of interference reduction in cellular networks studied in Chapter 18 is not based on network graphs, also the model considered there can in a way be regarded as a node-to-node interference model: Base stations choose their transmission ranges, which affect mobile nodes.

Certain interference models defined above do not reflect intuition of what interference is. Interference values in the models contained in the *out-sum* and the *in-all* subtrees can for instance reach $\Omega(n^2)$, n being the number of network nodes; such behavior can hardly be justified if—intuitively—a basic property of interference is that, considering a given node v, at most all other nodes can interfere with v or can be affected by v. The values in all other presented models cannot exceed n. Another family of questionably defined models is contained in the *out-avg* subtree. The interference value of a node is computed by averaging—over the number of incident edges—the sum of the cardinalities of all disks induced by incident edges, where the cardinality of a disk is the number of nodes it covers. With this measure, a node with a high interference value can reduce its interference by initiating connections to near-by nodes. That a node can diminish its interference by means of additional connections cannot be considered a reasonable property of an interference model. Nevertheless, the *out-avg* subtree is depicted in Figure 19.1 for illustration.

Furthermore, certain models yield identical interference values although having differing definitions. The I_{oma} and I_{ima} models produce identical values given the same graph. The reason for this identity is that every occurrence of a node covered by a disk is counted exactly once in both models, at the respective

interference originators in I_{oma} and at the nodes affected in I_{ima}. The same identity holds for the I_{osa} and I_{iaa} models.

As regards solvability of interference minimization problems in the discussed models, it appears that, in most cases, the measures based on incoming interference result in problems of more complex nature than those based on outgoing interference. Informally, one of the main reasons for this difference lies in the following observation: In an outgoing interference model, the interference caused by a node choosing an edge to be included in the resulting topology is a static property of that edge independent of the selection of other edges; in contrast, such a static correlation between interference values and single network entities does not exist in models with incoming interference. This apparently fundamental difference in complexity between problems based on outgoing and incoming interference can also be observed in the problems discussed in the previous chapters, although not all of these problems are based on node-to-node interference.

19.2 Edge-to-Edge Interference

A prominent example of a model where edges in the network graph are considered to interfere with other edges is [74]. Such a model can be justified by arguing that it is communication over links producing interference and thus preventing communication intended to take place over other links.

Like node-to-node interference, edge-to-edge interference models can be classified according to a set of parameters. The most fundamental such parameter is again the distinction between outgoing and incoming interference, counting how many other edges a given edge affects or by how many other edges a given edge is affected, respectively.

A parameter that only occurs in connection with edge-based interference models is edge symmetry. An edge can be considered to cause interference in one direction, that is around the sender only, or in both directions, around both incident nodes, according to how the question is answered whether communication over a link in one direction is inevitably (or forcedly) linked to communication in the opposite direction. Similarly, a link affected by interference can be considered either directed or symmetric in the following sense: On the one hand, interference can prevent reception only of a message, while the sending process of a message remains in principle unaffected by interfering signals; on the other hand it can be argued that communication over a link is always bidirectional and therefore an edge is interfered with if any of the incident nodes is affected. The interference model introduced in [74] for instance considers edges to be symmetric with respect to both causation and suffering of interference.

A final definition parameter is, as in the previous section, the question how to arrive, once the interference of a single edge is specified, at the interference of a whole graph. Again, standard options include maximizing and averaging over all edge interference values in the graph.

19.3 Edge-to-Node Interference

Another perspective on interference modeling is that edges interfere with nodes, as communication takes place over links and the network entities affected by interference are the nodes. The model studied in Chapter 15 is of this type (and also the model considered in Chapter 16 can be regarded as such).

Also the edge-to-node interference models can be defined with several parameters. Again, the principal choice is whether outgoing or incoming interference is to be considered, counting the nodes affected by an edge or the edges affecting a node. A second parameter is edge directionality, the question also posed in the previous section whether communication over an edge causes interference around both incident nodes or around the sender only. Finally, again for instance the maximum or average aggregation functions can be employed to compute network interference from node interference values. The model studied in Chapter 15, as an example, considers outgoing interference with symmetric edges and uses maximization to arrive at graph interference. Similarly, the interference definition discussed in Chapter 16 can be regarded as modeling incoming interference with asymmetric edges and maximizing over node interference values.

Comparing these two models, the complexity difference also mentioned at the end of Section 19.1 between problems based on outgoing interference and such based on incoming interference becomes particularly apparent. The problems discussed in Chapter 15 can be solved optimally with relatively simple greedy algorithms while the problems based on incoming interference in the subsequent chapters apparently have to be tackled using more complex approaches. It again appears that this difference in complexity is due to the fact that—for instance using maximization for the purpose of aggregation—with outgoing interference the resulting network interference value is attributed to a single node or edge causing that interference value, whereas no such correspondence of the resulting value with a single network entity exists if the considered measure is based on incoming interference.

19.4 Node-to-Edge Interference

Due to symmetry, also the fourth type of interference—nodes interfering with edges—is conceivable, arguing that network nodes, the entities sending signals and thus producing interference, prevent proper communication over links. Again, the models can be classified with respect to a set of parameters, particularly along the three parameters concerning outgoing or incoming interference, directionality of edges, and network-level interference pooling. The edges being the targets of interference, their directionality is considered to answer the question whether only the receiving node of a directed edge or both nodes incident to a symmetric edge can be affected by interference. Although this type of interference models can be justified, they are probably less intuitive than

those discussed in the previous sections; the node-to-edge interference type is therefore mentioned for completeness, mainly.

19.5 Concluding Remarks

After introducing a selection of interference models and studying interference reduction problems based on these models in the previous chapters, we intend in this chapter to allow the reader to take a look "behind the scenes", showing that a systematic exploration of the interference model "design space"—even if confined by the idealizing assumptions made at the beginning of the chapter—yields a host of possible measures, some of which can well be justified, while others contradict intuition, and of which those that can most reasonably be justified to reflect the notion of interference in wireless networks have been chosen for further analysis.

Chapter 20

Conclusion

A conclusion is the place where you got tired of thinking.
Martin H. Fischer (1879–1962)

Ad hoc and sensor network nodes are "unleashed" in two respects: They communicate using wireless connections, and they are independent of external power sources. The foremost consequences of this autonomy consist in potentially highly dynamic networks—caused by the inherent instability of radio links and by node mobility—and in energy constituting a critical resource. In the first part of this dissertation we analyzed geographic routing, a type of routing particularly promising for dynamic networks, mainly due to its strictly local nature. Topology control playing an important role already in this first part, the second part of the dissertation emphasized energy-aware topologies. In particular, interference was discussed as one of the crucial energy-consuming issues in ad hoc networks. The task of reducing interference among network nodes was approached by first identifying the notion of interference and subsequently proposing low-interference topologies.

Clearly, the discussed techniques, methods, and solutions are of heterogeneous maturity. Our geographic routing protocols—having been shown to be both optimal in worst-case networks and efficient on average—have reached a degree of design that suggests their being taken one stage further towards implementation in practical networks, even if some important issues with respect to node mobility have yet to be solved. On the other end of the spectrum, the exploration of interference-aware topology control algorithms may be considered but a first step on the long journey towards understanding interference in wireless ad hoc networks. The effect these considerations might have on practical networks is yet unclear. Above all an analytical characterization of the concept of signal-to-noise ratio, based on graph representations of networks, would help to move this approach forward. The lightweight topology control algorithm can be regarded as situated between these two extremes in the sense that it

not only features certain theoretical properties but also forms a reasonable basis for experimentation and exploration of interference and energy in practical networks.

At the end of this dissertation, it may be justified to ask what we can learn from this work. Technically, the geographic routing algorithms may stand as an example that it is possible to design algorithms with theoretically proved worst-case guarantees and more practically relevant average-case efficiency. Maybe, the general conclusion can be drawn that accounting for worst-case behavior before studying the average case appears to be easier than conversely, which may serve as a design principle beyond the scope of ad hoc and sensor networks. From a less technical perspective, the second part of the dissertation showed that sometimes questioning a seemingly trivial statement—low node degree implies low interference—can open up a new field full of fascinating problems, questions, and answers. Whether the main goal of this thesis, helping to narrow the gap between theory and practice, however has been reached, only time will show.

Bibliography

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a Locality Aware Location Service for Mobile Ad Hoc Networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of Mobile Computing*, pages 75–84, 2004.

[2] I. Akyildiz, W. Su, Y. Sanakarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks Journal*, 38(4):393–422, 2002.

[3] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proc. of the 3^{rd} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2002.

[4] B. Awerbuch. Complexity of Network Synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

[5] B. Awerbuch and D. Peleg. Network Synchronization with Polylogarithmic Overhead. In *Proc. of the 31^{st} IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 514–522, 1990.

[6] L. Bao and J. Garcia-Luna-Aceves. Topology Management in Ad Hoc Networks. In *Proc. of the 4^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[7] L. Barrière, P. Fraigniaud, and L. Narayanan. Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges. In *Proc. of the 5^{th} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 19–27. ACM Press, 2001.

[8] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proc. of the 4^{th} Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 76–84. ACM Press, 1998.

[9] R. Bischoff and R. Wattenhofer. Analyzing Connectivity-Based, Multi-Hop Ad-hoc Positioning. In *Proc. of the 2nd IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, Orlando, Florida, USA, 2004.

[10] D. Blough, M. Leoncini, G. Resta, and P. Santi. The K-Neigh Protocol for Symmetric Topology in Ad Hoc Networks. In *Proc. of the 4^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[11] S. Borbash and E. Jennings. Distributed Topology Control Algorithm for Multihop Wireless Networks. In *Proc. of the 2002 International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 355–360, May 2002.

[12] P. Bose, A. Brodnik, S. Carlsson, E. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J. Munro. Online Routing in Convex Subdivisions. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 1969 of *Lecture Notes in Computer Science*, pages 47–59. Springer, 2000.

[13] P. Bose and P. Morin. Online Routing in Triangulations. In *Proc. 10^{th} Int. Symposium on Algorithms and Computation (ISAAC)*, volume 1741 of Springer LNCS, pages 113–122, 1999.

[14] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In *Proc. of the 3^{rd} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 48–55, 1999.

[15] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J.Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[16] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does Topology Control Reduce Interference? In *Proceedings of the 5^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, pages 9–19, 2004.

[17] C. Busch, S. Surapaneni, and S. Tirthapura. Analysis of Link Reversal Routing Algorithms for Mobile Ad Hoc Networks. In *Proc. of the 15th Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, San Diego, California, USA, June 2003.

[18] E. Chang. Echo Algorithms: Depth Parallel Operations on General Graphs. *IEEE Transactions on Software Engineering*, pages 391–401, 1982.

[19] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks. In *Proc. IEEE Singapore Int. Conf. on Networks*, pages 197–211, 1997.

[20] C.-Y. Chong and S. P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[21] J. Chou, D. Petrovic, and K. Ramchandran. A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks. In *Proc. of the* 22^{nd} *Annual Joint Conference of the IEEE Computer and Comunications Societies (INFOCOM)*, 2003.

[22] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[23] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, 1983.

[24] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86:165–177, 1990.

[25] T. Clausen, Ed. and P. Jacquet, Ed. Optimized Link State Routing Protocol (OLSR). IETF RFC 3626, October 2003.

[26] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*, chapter 24.2. MIT Press, Cambridge, MA, USA, 1990.

[27] D. Estrin et al. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. The National Academies Press, Washington DC, 2001.

[28] S. Datta, I. Stojmenovic, and J. Wu. Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks. In *Cluster Computing 5*, pages 169–178. Kluwer Academic Publishers, 2002.

[29] O. Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *Proc. IEEE Infocom*, New York, NY, USA, June 2002.

[30] P. Doyle and J. Snell. *Random Walks and Electric Networks*. The Carus Mathematical Monographs. The Mathematical Association of America, 1984.

[31] H. Dubois-Ferrière, M. Grossglauser, and M. Vetterli. Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages. In *Proc. of the* 4^{th} *ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[32] D. Eppstein, Z. Galil, and G. Italiano. *CRC Handbook of Algorithms and Theory*, chapter Dynamic Graph Algorithms. CRC Press, 1997.

[33] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-Free Colorings of Simple Geometric Regions with Applications to Frequency Assignment in Cellular Networks. In *Proc. of the* 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[34] U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[35] U. Feige, M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating the Domatic Number. *SIAM Journal of Computing*, 32(1):172–195, December 2002.

[36] G. Finn. Routing and Addressing Problems in Large Metropolitan-scale Internetworks. Technical Report ISI/RR-87-180, USC/ISI, March 1987.

[37] M. Fussen, R. Wattenhofer, and A. Zollinger. Interference Arises at the Receiver. In *Proc. of the International Conference on Wireless Networks, Communications, and Mobile Computing (WirelessCom)*, June 2005.

[38] K. Gabriel and R. Sokal. A New Statistical Approach to Geographic Variation Analysis. *Systematic Zoology*, 18:259–278, 1969.

[39] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communication*, 29, 1981.

[40] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc.* 17^{th} *Annual Symposium on Computational Geometry (SCG)*, pages 188–196. ACM Press, 2001.

[41] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *Proc. ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2001.

[42] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *Proc. of the* 2^{nd} *ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, pages 45–55. ACM Press, 2001.

[43] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *Proc.* 2^{nd} *ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, USA, October 2001.

[44] A. Goel and D. Estrin. Simultaneous Optimization of Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[45] M. Grossglauser and M. Vetterli. Locating Nodes with EASE: Mobility Diffusion of Last Encounters in Ad Hoc Networks. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[46] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. In J. Díaz and M. Serna, editors, *Algorithms—ESA '96, Fourth Annual European Symposium*, volume 1136 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1996.

[47] T. Hou and V. Li. Transmission Range Control in Multihop Packet Radio Networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[48] L. Hu. Topology Control for Multihop Packet Radio Networks. *IEEE Trans. on Communications*, 41(10), 1993.

[49] H. Huang, A. W. Richa, and M. Segal. Approximation Algorithms for the Mobile Piercing Set Problem with Applications to Clustering in Ad-hoc Networks. In *Proc. of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 52–61, 2002.

[50] H. Hunt III, M. Marathe, V. Radhakrishnan, S. Ravi, D. Rosenkrantz, and R. Stearns. NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. *J. Algorithms*, 26(2):238–274, 1998.

[51] J. Janssen. Channel Assignment and Graph Labeling. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 5, pages 95–117. John Wiley & Sons, Inc., 2002.

[52] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[53] M. Joa-Ng and I.-T. Lu. A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks. *IEEE Journal on Selected Areas In Communication*, 17(8):1415–1425, August 1999.

[54] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[55] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[56] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica 4*, pages 373–395, 1984.

[57] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. 6^{th} Annual Int. Conf. on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[58] V. Kawadia and P. Kumar. Power Control and Clustering in Ad Hoc Networks. In *Proc. of the 22^{nd} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[59] Y.-B. Ko and N. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proc. of the 4^{th} Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, 1998.

[60] Y.-B. Ko and N. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. In *Proc. 2^{nd} IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.

[61] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing on Geometric Networks. In *Proc. 11^{th} Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[62] P. Krishna, M. Chatterjee, N. H. Vaidya, and D. K. Pradhan. A Cluster-based Approach for Routing in Ad-Hoc Networks. In *Proc. 2^{nd} USENIX Symposium on Mobile and Location-Independent Computing*, pages 1–10, 1995.

[63] F. Kuhn, P. von Rickenbach, R. Wattenhofer, E. Welzl, and A. Zollinger. Interference in Cellular Networks: The Minimum Membership Set Cover Problem. In *Proc. of the* 11^{th} *International Computing and Combinatorics Conference (COCOON)*, Kunming, Yunnan, China, August 2005.

[64] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22^{nd} ACM Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[65] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Routing: Of Theory and Practice. In *Proc. of the 22^{nd} ACM Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[66] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. In *Proc. 6^{th} Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial-M)*, pages 24–33. ACM Press, 2002.

[67] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-Hoc Networks Beyond Unit Disk Graphs. In *1^{st} ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, San Diego, California, USA, September 2003.

[68] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In *Proc. 4th ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[69] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, Aug. 2000.

[70] N. Li, C.-J. Hou, and L. Sha. Design and Analysis of an MST-Based Topology Control Algorithm. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[71] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed Construction of Planar Spanner and Routing for Ad Hoc Wireless Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[72] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5):960–981, 1994.

[73] R. Meester and R. Roy. *Continuum Percolation*. Cambridge University Press, Cambridge, 1996.

[74] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, Congestion and Dilation in Radio Networks. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures*, 10 - 13 Aug. 2002.

[75] K. Moaveni-Nejad, W.-Z. Song, W.-Z. Wang, and X.-Y. Li. Low-Interference Topology Control for Wireless Ad-hoc Networks. In *Proceedings of the 1st Int. Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN)*, 2004.

[76] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual Coordinates for Ad hoc and Sensor Networks. In *Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, Philadelphia, Pennsylvania, USA, 2004.

[77] T. Moscibroda and R. Wattenhofer. Maximizing the Lifetime of Dominating Sets. In *Proc. of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Denver, Colorado, USA, April 2005.

[78] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[79] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *Proc. of Information Processing in Sensor Networks (IPSN)*, 2003.

[80] L. Narayanan. Channel Assignment and Graph Multicoloring. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 4, pages 71–94. John Wiley & Sons, Inc., 2002.

[81] J. Navas and T. Imielinski. GeoCast - Geographic Addressing and Routing. In *Proc. 3^{rd} Int. Conf. on Mobile Computing and Networking (MobiCom)*, pages 66–76, 1997.

[82] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *Proc. of IEEE Global Communications (GLOBECOM)*, 2001.

[83] N. Nikaein, C. Bonnet, and N. Nikaein. HARP - Hybrid Ad Hoc Routing Protocol. In *IST 2001, International Symposium on Telecommunications, Teheran, Iran*, Sep 2001.

[84] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). IETF RFC 3684, February 2004.

[85] V. Park and M. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. of the 16^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1405–1413, 1997.

[86] D. Peleg. *Distributed Computing, A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[87] D. Peleg and A. Schäffer. Graph Spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[88] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[89] C. Perkins and E. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proc. of the 2^{nd} IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[90] C. E. Perkins. *Mobile IP: Design Principles and Practices*. Prentice Hall PTR, 1998.

[91] C. E. Perkins. Mobile Networking Through Mobile IP. *IEEE Internet Computing*, 2(1):58–69, January/February 1998.

[92] R. Prakash. Unidirectional Links Prove Costly in Wireless Ad-Hoc Networks. In *Proc. of the 3^{rd} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 1999.

[93] P. Raghavan and C. Thompson. Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica*, 7(4):365–374, 1987.

[94] R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proc. of the 19^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[95] V. Ramasubramanian, Z. Haas, and E. Sirer. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In *Proc. of the 4^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[96] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica. Geographic Routing without Location Information. In *Proceedings of the 9^{th} ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[97] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. *IEEE J. Selected Areas in Communications*, 17(8), 1999.

[98] E. Royer and C. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. In *IEEE Personal Communications*, volume 6, April 1999.

[99] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc networks of Sensors. In *Proceedings of the 7^{th} ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2001.

[100] A. Srinivasan. Improved Approximations of Packing and Covering Problems. In *Proc. of the 27th ACM Symposium on Theory of Computing (STOC)*, pages 268–276, 1995.

[101] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[102] G. Toussaint. The Relative Neighborhood Graph of a Finite Planar Set. *Pattern Recognition*, 12(4):261–268, 1980.

[103] J. Urrutia. Routing with Guaranteed Delivery in Geometric and Wireless Networks. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 18, pages 393–406. John Wiley & Sons, 2002.

[104] P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger. A Robust Interference Model for Wireless Ad-Hoc Networks. In *Proc. of the 5^{th} International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Denver, Colorado, USA, April 2005.

[105] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, Philadelphia, Pennsylvania, USA, October 2004.

[106] Y. Wang and X.-Y. Li. Distributed Spanner with Bounded Degree for Wireless Ad Hoc Networks. In *2^{nd} International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2002.

[107] Y. Wang and X.-Y. Li. Geometric Spanners for Wireless Ad Hoc Networks. In *Proc. 22^{nd} International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[108] Y. Wang and X.-Y. Li. Localized Construction of Bounded Degree Planar Spanner. In *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2003.

[109] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *Proc. of the 20^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1388–1397, 2001.

[110] R. Wattenhofer and A. Zollinger. XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks. In *Proc. of the 4^{th} Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2004.

[111] J. Wu. Dominating-Set-Based Routing in Ad Hoc Wireless Networks. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 20, pages 425–450. John Wiley & Sons, 2002.

Curriculum Vitae

April 24, 1975	Born in Schönenwerd SO, Switzerland
1982–1996	Rudolf Steiner primary, secondary, and high schools in Schafisheim AG, Adliswil ZH, and Zurich, Switzerland
1996–2001	Studies in computer science, ETH Zurich, Switzerland
April 2001	Diploma in computer science, ETH Zurich, Switzerland
2001–2005	Ph.D. student, research and teaching assistant, Distributed Computing Group, Prof. Roger Wattenhofer, ETH Zurich, Switzerland
April 2005	Ph.D. degree, Distributed Computing Group, ETH Zurich, Switzerland Advisor: Prof. Roger Wattenhofer Co-examiners: Prof. Matthias Grossglauser, EPFL Lausanne, Switzerland Charles E. Perkins, Nokia Research, Mountain View, CA, USA