UNIVERSITÄT KOBLENZ · LANDAU

unikorn Computer Networks Group

Silence is Golden: Reactive Local Topology Control and Geographic Routing in Wireless Ad Hoc and Sensor Networks

Dissertation

vorgelegt von

Florentin Neumann, M.Sc.

Koblenz, August 2016

Vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation

Vorsitzender des Promotionsausschusses
Prof. Dr. Ralf Lämmel, Universität Koblenz-Landau

Promotionskommission

Prof. Dr. Viorica Sofronie-Stokkermans, Universität Koblenz-Landau	Vorsitzende
Prof. Dr. Hannes Frey, Universität Koblenz-Landau	Berichterstatter
Prof. Dr. Volker Turau, Technische Universität Hamburg-Harburg	Berichterstatter

Datum der Einreichung	30. März 2016
Datum der wissenschaftlichen Aussprache	13. Juli 2016

Abstract

Reactive local algorithms are distributed algorithms which suit the needs of battery-powered, large-scale wireless ad hoc and sensor networks particularly well. By avoiding both unnecessary wireless transmissions and proactive maintenance of neighborhood tables (i.e., beaconing), such algorithms minimize communication load and overhead, and scale well with increasing network size. This way, resources such as bandwidth and energy are saved, and the probability of message collisions is reduced, which leads to an increase in the packet reception ratio and a decrease of latencies.

Currently, the two main application areas of this algorithm type are *geographic routing* and *topology control*, in particular the construction of a node's adjacency in a connected, planar representation of the network graph. Geographic routing enables wireless multi-hop communication in the absence of any network infrastructure, based on geographic node positions. The construction of planar topologies is a requirement for efficient, local solutions for a variety of algorithmic problems.

This thesis contributes to reactive algorithm research in two ways, on an *abstract level*, as well as by the introduction of *novel algorithms*:

For the very first time, reactive algorithms are considered as a whole and as an individual research area. A comprehensive survey of the literature is given which lists and classifies known algorithms, techniques, and application domains. Moreover, the mathematical concept of $\mathcal{O}$- and Ω-*reactive local topology control* is introduced. This concept unambiguously distinguishes reactive from conventional, beacon-based, topology control algorithms, serves as a taxonomy for existing and prospective algorithms of this kind, and facilitates in-depth investigations of the principal power of the reactive approach, beyond analysis of concrete algorithms.

Novel reactive local topology control and geographic routing algorithms are introduced under both the *unit disk* and *quasi unit disk graph model*. These algorithms compute a node's local view on connected, planar, constant stretch Euclidean and topological spanners of the underlying network graph and route messages reactively on these spanners while guaranteeing the messages' delivery. All previously known algorithms are either not reactive, or do not provide constant Euclidean and topological stretch properties. A particularly important partial result of this work is that the *partial Delaunay triangulation* (PDT) is a constant stretch Euclidean spanner for the unit disk graph.

To conclude, this thesis provides a basis for structured and substantial research in this field and shows the reactive approach to be a powerful tool for algorithm design in wireless ad hoc and sensor networking.

Zusammenfassung

Reaktiv lokale Algorithmen sind verteilte Algorithmen, die den Anforderungen großer, batteriebetriebener, Drahtloser Ad Hoc und Sensornetzwerke im besonderen Maße gerecht werden. Durch Vermeidung überflüssiger Nachrichtenübertragungen sowie Verzicht auf proaktive Ermittlung von Nachbarschaftstabellen (d.h. *beaconing*) minimieren solche Algorithmen den Kommunikationsaufwand und skalieren gut bei wachsender Netzgröße. Auf diese Weise werden Ressourcen wie Bandbreite und Energie geschont, es kommt seltener zu Nachrichtenkollisionen und dadurch zu einer Erhöhung der Paketempfangsrate, sowie einer Reduktion der Latenzen. Derzeit wird diese Algorithmenklasse hauptsächlich für *Geografisches Routing*, sowie zur *Topologiekontrolle*, insbesondere zur Ermittlung der Adjazenzliste eines Knotens in zusammenhängenden, kantenschnittfreien (planaren) Repräsentationen des Netzgraphen, eingesetzt. Ersteres ermöglicht drahtlose multi-hop Kommunikation auf Grundlage von geografischen Knotenpositionen ohne Zuhilfenahme zusätzlicher Netzwerkinfrastruktur, wohingegen Letzteres eine hinreichende Grundlage für effiziente, lokale Lösungen einer Reihe algorithmischer Problemstellungen ist.

Die vorliegende Dissertation liefert neue Erkenntnisse zum Forschungsgebiet der reaktiven Algorithmen, zum Einen auf einer *abstrakten Ebene* und zum Anderen durch die Einführung *neuer Algorithmen*.

Erstens betrachtet diese Arbeit reaktive Algorithmen erstmalig im Ganzen und als eigenständiges Forschungsfeld. Es wird eine umfangreiche Literaturstudie zu dieser Thematik präsentiert, welche die aus der Literatur bekannten Algorithmen, Techniken und Anwendungsfelder systematisch auflistet, klassifiziert und einordnet. Weiterhin wird das mathematische Konzept der $\mathcal{O}$- und Ω-*reaktiv lokalen Topologiekontrolle* eingeführt. Dieses Konzept ermöglicht erstmals die eindeutige Unterscheidung reaktiver von konventionellen, beacon-basierten, verteilten Topologiekontrollalgorithmen. Darüber hinaus dient es als Klassifikationsschema für existierende, sowie zukünftige Algorithmen dieser Art. Zu guter Letzt ermöglicht dieses Konzept grundlegende Aussagen über die Mächtigkeit des reaktiven Prinzips, welche über Entwurf und Analyse von Algorithmen hinaus reichen.

Zweitens werden in dieser Arbeit neue reaktiv lokale Algorithmen zur Topologiekontrolle und Geografischem Routing eingeführt, wobei drahtlose Netze durch *Unit Disk* bzw. *Quasi Unit Disk Graphen* modelliert werden. Diese Algorithmen berechnen für einen gegebenen Knoten die lokale Sicht auf zusammenhängende, planare, Euklidische bzw. Topologische Spanner mit konstanter Spannrate bzgl. des Netzgraphen und routen Nachrichten reaktiv entlang der Kanten dieser Spanner, wobei die Nachrichtenauslieferung garantiert wird. Alle bisher bekannten Verfahren sind entweder nicht reaktiv oder gewährleisten keine konstanten Euklidischen oder Topologischen Spannraten. Ein wesentliches Teilergebnis dieser Arbeit ist der Nachweis, dass die *partielle Delaunay Triangulierung* (PDT) ein Euklidischer Spanner mit konstanter Spannrate für Unit Disk Graphen ist.

Die in dieser Dissertation gewonnenen Erkenntnisse bilden die Basis für grundlegende und strukturierte Forschung auf diesem Gebiet und zeigen, dass das reaktive Prinzip ein wichtiges Werkzeug des Algorithmenentwurfs für Drahtlose Ad Hoc und Sensornetzwerke ist.

Acknowledgements

I am indebted to several people who have contributed to this work—by one means or another—and to whom I would like to express my gratitude.

A sincere thanks is owed to my supervisor Hannes Frey. His guidance and support, as well as the many inspiring past and ongoing research discussions have provided the basis for this thesis. Whenever I have a question, even at the busiest of times, I literally find Hannes' office door open. I do not take this for granted.

Furthermore, I would like to acknowledge the effort and time spent by my co-examiner Volker Turau. His positive and constructive feedback helped to improve this work. I appreciate that I was given the opportunity to present my research at the Institute of Telematics at the Hamburg University of Technology.

Let me also express my special thanks to Ivan Stojmenović. This thesis cites more than two dozen of his publications and heavily relies on his outstanding research. His interest in my research findings at conferences and during his visits to Koblenz honored me greatly. May he rest in peace.

My present and former colleagues of the *Unikorn*-team here in Koblenz, namely, Rafael, Jovan, Frank, and Sabine deserve special mention. Thanks for the everyday fun, support, and enlightening talks, not only about research, but on almost any aspect of life on earth. The same applies to the Computer Networks Group at the University Paderborn led by Holger Karl, who I had the pleasure to work with for the first one and a half years.

Of course, I am grateful to all the students I advised in their theses as well as all the student assistants that contributed to the research project "Reactive Spanner". This project is funded by the German Research Foundation (DFG) under grants FR 2978/1-1 and FR 2978/1-2 and has constituted the financial basis of this work.

Lastly, I owe heartfelt thanks to two particularly important people. I would like to express my deep gratitude to my mother, who has always supported me wholeheartedly. My final and special word of thanks is dedicated to Janine: Thank you for your everyday and unconditional support. Your are an inspiring example to me!

Koblenz, August 2016

Contents

Glossary of Notations

This is a summary of the most frequently used notations.

Let $u, v, w \in \mathbb{R}^2$ be three distinct and non-collinear points, $V \subset \mathbb{R}^2$ a distinct and finite point set, and g, f two functions defined on some subset of the real numbers.

Elementary Geometric Concepts

$\mathrm{x}(u)$, $\mathrm{y}(u)$	x- and y-component of u
$\|uv\|$	Euclidean distance between u and v
$\ell(u, v)$	Straight line passing through u and v
$\overline{uv}$	Closed line segment betwen u and v
$\mathrm{D}(u, v)$	Closed disk corresponding to the circle whose diameter is $\overline{uv}$
$\mathrm{C}_\mathrm{R}(u)$	Circle centered at u with radius R
$\mathrm{D}^\mathrm{o}_\mathrm{R}(u)$	Open disk centered at u with radius R
$\mathrm{C}(u, v, w)$	Unique circle defined by u, v, and w
$\mathrm{D}^\mathrm{o}(u, v, w)$	Open disk defined by circle $\mathrm{C}(u, v, w)$
d_{uvw}	Diameter of circle $\mathrm{C}(u, v, w)$
$\triangle(u, v, w)$	Triangle defined by u, v, and w
$\measuredangle uvw$	Internal angle at v of triangle $\triangle(u, v, w)$
$\mathrm{B}(u, v)$	Perpendicular bisector of line segment $\overline{uv}$
$\mathrm{H}(u, v)$	Closed half-plane defined by $\mathrm{B}(u, v)$ and containing u
$\mathrm{H}^w(u, v)$	Open half-plane defined by $\ell(u, v)$ and containing w

Geometric Graphs and Computational Geometry

$\mathrm{UDG}(V)$	Unit disk graph over set V (see Definition 2.1)
$\mathrm{QUDG}(V)$	Quasi unit disk graph over set V (see Definition 2.2)
$\mathrm{VD}(V)$	Voronoi diagram of V (see Definition 2.16)
$\mathrm{VR}_V(u)$	Voronoi region of $u \in V$ w.r.t. $\mathrm{VD}(V)$ (see Definition 2.16)
$\mathrm{Del}(V)$	Delaunay triangulation over V (see Definition 2.17)
$\mathrm{RNG}(V)$	Relative neighborhood graph over V (see Definition 2.19)
$\mathrm{GG}(V)$	Gabriel graph over V (see Definition 2.20)
$\mathrm{PDT}(V)$	Partial Delaunay triangulation over V (see Definition 3.1)
$\mathrm{N}^G_k(u)$	Set of nodes reachable from node u via k hops in graph G including u itself (superscript G is ommitted if it is clear from context)

Asymptotic Growth of Functions [1]

$\mathcal{O}(g(n)) = \{f(n) : \exists\, c > 0, n_0 > 0 \text{ s.t. } 0 \leq f(n) \leq c \cdot g(n), \forall\, n \geq n_0\}$
$\Omega(g(n)) = \{f(n) : \exists\, c > 0, n_0 > 0 \text{ s.t. } 0 \leq c \cdot g(n) \leq f(n), \forall\, n \geq n_0\}$
$\Theta(g(n)) = \{f(n) : \exists\, c_1, c_2 > 0, n_0 > 0 \text{ s.t. } 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \forall\, n \geq n_0\}$

Chapter 1

Introduction

Wireless ad hoc networks are decentralized networks that do not require any fixed network infrastructure. Network nodes communicate directly using wireless transceivers. Due to restricted hardware and energy resources, the network nodes' communication ranges are limited. Communication between distant nodes is achieved via multi-hop routing, where each node simultaneously acts as a router as well as an end system. Wireless ad hoc networks are intended for deployment in areas where it is impossible or undesirable to set up wireless networking infrastructures as required, e.g., by *cellular* or *wireless local area networks*.

Wireless sensor networks are ad hoc networks, where the network nodes are small and low-cost computers, equipped with additional sensing capabilities. Such networks are designed for fast and easy access to physical world data and phenomena and have a wide range of applications such as environmental monitoring (see [2–5] for more examples).

Bandwidth and energy are scarce resources in such networks [6]. The wireless transmission of a message is a particularly costly operation regarding both of these resources. Completing tasks with as few message transmissions as possible is rewarding: it saves resources and alleviates typical wireless networking problems such as message collisions and interference. To put it in a nutshell: *Silence is golden* in wireless ad hoc and sensor networks [7, 8]. However, the inherent purpose of such networks is—of course—the communication of data. Therefore, this principle must be understood as replacing unnecessary transmissions and communication overhead by silence.

This is the rationale of *reactive* (aka *beaconless*) algorithms. They consistently avoid unnecessary message transmissions and communication overhead: Unless there is an event to handle, the network nodes keep silent and do not even determine their immediate network neighborhoods. In the case of an event, using "blind" local broadcasts and contention mechanisms, exactly those nodes are requested to actively participate by sending of messages, which are actually necessary to handle this event. All other nodes remain silent. This way, in total fewer messages are transmitted compared to conventional distributed approaches.

The overall objective of this thesis is the advancement of the reactive approach and reactive algorithms research as a whole, as well as the design of novel reactive algorithms for topology control and geographic routing in wireless ad hoc and sensor networks.

This objective is motivated in detail and further specified in Sections 1.1 & 1.2. Section 1.3 summarizes the contributions of this thesis. In Sections 1.4 & 1.5, own publications used in this work are listed and an outline of this thesis is given.

1.1 Motivation

Scalability and *resource efficiency* are two major challenges in realizations of wireless ad hoc and sensor networks [3, 6] and have to be taken into account in algorithm design.

Existing networks already consist of thousands of nodes [9] and perspectively this number could significantly increase in the future [10]. Therefore, algorithms for such networks have to scale well with increasing network size in terms of number of nodes.

Resource efficiency is of importance due to the limited resources, in particular regarding energy and network bandwidth [6]. Common approaches to conserve energy resources of nodes are, e.g., to operate the nodes' transceivers at *low duty cycle*, i.e., to turn off the transceivers while they are not required [2], and *dynamic adaption of transmit power* (see [11] for a survey on energy saving strategies). Bandwidth can be saved by reducing the overall communication load on the network. This can be achieved by reducing communication overhead and improving the *message efficiency* of the algorithms in use. The latter can be measured, for example, as the total number of message transmissions required for a specific operation.

Increasing message efficiency of operations, and more generally, reducing the overall communication load on the network is rewarding: A lesser number of message transmissions helps to reduce interference in the network, decreases the probability of collisions, and therefore, aids in increasing the packet reception ratio and to lowering latencies.

At least with respect to message efficiency and scalability, the use of *local algorithms* is predestined in wireless ad hoc and sensor networks. A distributed algorithm is said to be *k-local* if its execution by a node relies on information about the network topology that can be gathered from those nodes that are at most k hops apart from it. An algorithm is said to be *local* if it is k-local for some constant k. Local algorithms scale well with increasing network size because topological changes in the network, such as (dis)appearances of nodes and communication links between them, affect only nodes in the proximity of this change. They do not have to be communicated across the entire network, which would impose significant costs regarding the number of message transmissions in the order of the network's size.

In local algorithm design it is typically assumed that nodes are provided with complete k-hop neighborhood information, for some constant $k \geq 1$. That is, it is assumed that nodes know the addresses (geographic positions) and possibly adjacencies of all nodes that are at most k hops apart from them in the network graph. For a node to know its entire one-hop neighborhood information presupposes at least that each such neighbor has announced its existence via transmission of some "*HELLO*"-message, called a *beacon*. The periodic sending of such beacons for maintaining nodes' neighborhood information is referred to as *beaconing* [4]. Given that each node is provided with its one-hop neighborhood information, it is easy to provide nodes with their complete k-hop neighborhood information, for $k > 1$, by letting them successively announce their i-hop neighborhoods for $i = 1..k-1$. However, beaconing and in particular gathering of full k-hop neighborhood information, for $k > 1$, by all network nodes is a problematic process. Transmission, reception, and processing of beacons consumes additional energy. These transmissions interfere with regular data transmissions and therefore, increase the number

of collisions, which causes additional retransmissions. In addition, beaconing introduces additional control overhead, which reduces the available bandwidth for user data [12]. Lastly, gathering full neighborhood information takes a long time. Especially in highly dynamic networks, beaconing often fails to provide a node with current and correct neighborhood information, which is undesirable, because inaccurate neighborhood tables are the main source of packet loss in uncongested networks [12].

An additional key observation regarding message efficiency of local algorithms is that maintenance of complete neighborhood information by the nodes is simply superfluous for solving certain problems. For instance, in the case of local routing, it is absolutely sufficient if the node currently holding the packet determines the existence of exactly one suitable next-hop neighbor. Beacons transmitted by less or even unsuitable neighbors can be saved without losing relevant information for the routing decision, which positively affects the message efficiency of the routing process.

The above observations have recently led to the research field of *reactive algorithms*[1] [14] (also referred to as *beaconless* [16] or *contention-based* [17] algorithms), which is the primary object of study of this thesis.

Reactive algorithms are characterized by the following principles. Algorithm execution can be started by any node instantaneously at any given point in time. In particular, the execution neither includes nor relies on any proactive component such as beaconing. Hence, on algorithm start, the executing node is completely unaware of its network neighborhood or any other topological information except for its local node knowledge such as its address (typically its geographic position). Reactive algorithms take advantage of the wireless channel's property that the transmission of a message from a node to all of its neighbors does not presuppose any knowledge of them. Therefore, the algorithm execution starts with a "blind" local broadcast by the executing node, possibly containing parts of its local node knowledge. Based on this information and their own local knowledge, neighboring nodes overhearing the transmission either decide to stay passive and not to respond at all, or to join a timer-based contention. During this contention, nodes that are more eligible to solve the problem at hand respond earlier than others. Based on overheard earlier responses, some nodes resign from the contention and also become passive. Ultimately, only a few eligible neighbors have responded. For this reason, reactive approaches are advantageous compared to conventional, beacon-based approaches regarding message efficiency, because in beacon-based approaches neighbors always transmit a message, regardless of their eligibility.

Adherence to the reactive principle improves message efficiency (by reducing the total number of message transmissions, as well as by avoidance of control overhead imposed by beaconing) and helps to reduce the energy consumption of nodes in comparison to conventional, beacon-based approaches. Apart from theoretical worst-case scenarios, this has been shown empirically in various simulation studies (see e.g., [18–22], just to name a few), as well as in actual testbed experiments (see e.g., [23–25]).

[1] In literature on algorithmic aspects of ad hoc and sensor networking, an algorithm is often simply said to be *reactive* if it can be executed *on demand*, e.g., in [13]. It is important to note that the term *reactive algorithm* as used previously in [14, 15] as well as in this thesis is more meaningful and, hence, should not be confused with the aforementioned meaning.

Nevertheless, it is important to note that reactive algorithms are not per se more energy efficient than conventional distributed algorithms that make use of complete neighborhood information. At least for small transmit powers, the *transmit* and *receive modes* of a transceiver consume power in the same order of magnitude [3]. Although reactive algorithms often significantly reduce the number of message transmissions, during the contention phase the nodes' transceivers still have to operate in receive mode in order to facilitate overhearing; hence, considerable amounts of energy are consumed.

However, there are numerous examples [24,26–34] which show that the reactive approach can be further complemented with the aforementioned energy-saving techniques such as duty-cycling and dynamic transmit power adaption to further improve the energy efficiency. On top of that, reactive algorithms distinguish themselves by reducing the overall communication load on the network, which is by itself a highly desirable goal for the reasons given above.

The two main application areas of the reactive approach are currently *topology control* and *geographic routing*.

Topology control refers to the transformation of a given network graph into a graph (target topology) which satisfies certain desirable properties. A topology algorithm is local if it is a k-local algorithm for some constant k. The task of a *reactive topology control algorithm* is to generate a node's local view on its incident edges in the target topology on demand and without prior determination of this node's neighborhood in the network graph.

There are various aspects and goals of topology control algorithms (see [3, 6] for summaries). In the context of wireless sensor networks, *connectivity* and *stretch factor* are among the most important characteristics of topology control algorithms, apart from low computational and communication overhead [3].

First of all, a topology control algorithm should not disconnect a formerly connected network graph, but should always maintain the graph's *connectivity*. Secondly, it should not stretch pairwise node distances too much. Given a connected graph $G = (V, E)$ and a connected, spanning subgraph $H = (V, E' \subset E)$ of G, the *stretch factor* (or *spanning ratio*) of H w.r.t. G refers to the maximum factor by which the length of the shortest path (measured, e.g., in terms of number of hops or Euclidean length) between any two nodes in G is stretched compared to the shortest path in H. If the stretch factor is bounded by a constant, H is referred to as a *spanner*. A spanner is referred to as a *topological spanner* if path length is measured in terms of number of hops. If G is a network graph embedded in the plane and the length of a path is defined by the sum of the Euclidean lengths of its constituting edges, then a spanner is referred to as a *Euclidean spanner*.

Many algorithmic challenges in wireless ad hoc and sensor networking can be solved efficiently by means of local and localized algorithms[2], provided that the network graph is *planar*. Given an embedding of the network graph in the plane, a graph is said to be planar if in its straight line drawing, edges intersect only in their endpoints (see Definition 2.9).

[2] For a concise distinction between these terms see Section 2.2.

Given a connected and planar network graph, the following problems can be efficiently solved using known local(ized) algorithms:

- Guaranteed delivery geographic *routing* [35–38], *multicast* [39–42]), *geocast* [43], *anycast* [44], *mobicast* [45], and *broadcast* [46, 47]
- Communication *void* and *boundary detection* [48]
- Degree-limited localized bluetooth *scatternet formation* [49, 50]
- Efficient construction of *geographic overlays* [51]
- *Distributed data storage* based on geographic hashing [52, 53]
- *Tracking of mobile objects* by nearest sensor nodes [54]
- Localized *address autoconfiguration* [55]
- *Coordination* of mobile sensors [56]

The problem is that wireless network graphs are typically not planar. Therefore, to solve these problems correctly in a local manner, the nodes require local topology control algorithms for computation of their adjacencies in planar and connected topologies. In addition, for all of the aforementioned problems and algorithmic solutions, Euclidean or topological spanners are strongly preferred: (1) All of the routing applications [35–47] can route a message along a short path, consisting of a few hops or covering a short Euclidean distance, only if such a path is guaranteed to exist. (2) The principle of node aggregation, used in [51] for the construction of geographic overlays, approximates shortest path distances well, only if the planar topology over which it is constructed does so. (3) In localized address autoconfiguration [55], the use of planar topological spanners leads to fewer addressing agents. (4) When using planar Euclidean spanners in [54], closer approximations of the Euclidean distance to the tracked object can be obtained.

In order to reduce the costs (in terms of number of message transmissions and control overhead) incurred by such topology control operations, it is strongly desirable to design topology control algorithms that are both local and reactive.

It can be summarized that the reactive local construction of planar Euclidean and topological spanners has various applications and therefore, research on this subject is of actual importance. However, the current state of research regarding this problem statement is deficient. On the one hand, there are local algorithms (surveyed in Chapter 3) for construction of planar, constant stretch Euclidean and topological spanners, but these algorithms are not reactive. On the other hand, there are reactive topology control algorithms (surveyed in Chapter 4) for construction of planar graphs, but these graphs are not constant stretch Euclidean or topological spanners. Furthermore, there is no coherent conception of the class of reactive topology control algorithms. This circumstance not only complicates comparison of known and prospective results, but also hinders the derivation of fundamental propositions in this research field that go beyond algorithm design and analysis. These observations are the motivation for the first two research objectives of this thesis, which are presented shortly.

Geographic routing is the other central object of investigation in reactive algorithms research. Generally, geographic routing refers to the step-by-step forwarding process of a message from a source to a destination, based on geographic node positions. It presupposes that nodes are provided with their geographic positions and that the source is provided with a priori knowledge on the destination's geographic position. A deterministic geographic routing algorithm is said to *guarantee delivery*, if the packet always reaches its destination after a finite number of routing steps; it is said to be *local* if it is a k-local algorithm for some constant k; and it is *stateless*, if at most a constant number of node addresses are allowed to be stored in the packet header and nodes do not have to store previous routing decisions. In *reactive geographic routing*, the forwarding decision of a node has to be taken on demand and without determination of this node's neighborhood in the network graph.

Known local and stateless geographic routing algorithms that guarantee message delivery are based on the ideas of planar graph routing, such as FACE routing [36, 57] (see Section 2.3 for details). In order to guarantee the message delivery in case of a network void, local topology control is used to route the packet along faces of a planar subgraph of the underlying network graph. To avoid unnecessary routing hops and detours, it is desirable that the planar graph provides constant topological and Euclidean stretch properties, respectively.

Concerning reactive geographic routing, the strategy to first determine a node's local view on a connected, planar, constant stretch spanner, and then to select the next forwarding edge based on this local view is not message-efficient. It is much more efficient to determine reactively only the most suitable forwarding edge, as this saves unnecessary message transmissions. However, as in the case of topology control, the state of research in geographic routing is incomplete. On the one hand, there are guaranteed delivery, local geographic routing algorithms that route packets along edges of constant stretch Euclidean and topological spanners, but these algorithms are not reactive. On the other hand, all existing guaranteed delivery, reactive geographic routing protocols use edges of graphs whose Euclidean and topological stretch factors are either unknown, or known to be non-constant. This research gap constitutes the third of the three main objectives of this thesis, which are presented next.

1.2 Research Objectives

Based on the above motivation, in particular the gaps and open questions in the current state of research specified therein, the three main objectives and research questions of this thesis can be summarized as follows.

Objective 1: Foundation of reactive local topology control The goal is to generalize and formalize the class of reactive topology control algorithms as well as the underlying problem statement. These formalizations should

- unambiguously distinguish reactive from conventional, beacon-based, distributed topology control algorithms,

- serve as taxonomy for all existing and prospective algorithms of this kind, and
- facilitate in-depth investigations of the general power of the reactive approach.

Objective 2: Reactive algorithms for construction of planar spanners Based on commonly used and appropriate model assumptions of wireless ad hoc and sensor networks, the goal is to design mathematically correct, reactive local topology control algorithms, for construction of a node's local view (adjacency) on connected, planar, constant stretch spanners of the underlying network graph.

Objective 3: Reactive geographic routing on planar spanners Based on commonly used and appropriate model assumptions of wireless ad hoc and sensor networks, the goal is to design mathematically correct, guaranteed delivery reactive geographic routing algorithms that route the message along edges of a constant stretch spanner, without prior construction of each node's local view on this spanner.

The spanner property addressed in Objectives 2 & 3 should be achieved in terms of topological or Euclidean path length for the reasons given in the motivation.

There are numerous models for wireless ad hoc and sensor networks. Regarding algorithmic aspects of such networks, the most prevalent models are the *unit disk* or *quasi unit disk graph*. For reasons of simplicity and compatibility with prior work in this area, these models shall be used for the above research questions. For a more detailed discussion and additional arguments in favor of this choice, see Section 2.1.

1.3 Summary of Contributions

In the following, the contributions of this thesis are listed in their order of appearance. A discussion to which extent these contributions match the afore-stated research objectives is part of the conclusion in Chapter 10.

Survey on local(ized) planarization techniques and spanner construction - Chapter 3 This survey systematically and comprehensively reviews local and localized techniques for planarization and construction of planar spanners under unit disk and quasi unit disk graph model assumptions. For both graph classes, the individual approaches are categorized into groups with common modes of operation. Due to the quantity of publications dealing with unit disk graph planarization, only those approaches which actually output planar graphs are listed. There are fewer publications dealing with quasi unit disk graphs. This allows to present also those approaches that construct spanners which are not necessarily planar. A tabular taxonomy for both graph classes is provided that enables comparison of the individual approaches.

Although there are several survey papers, textbooks, and textbook chapters on spanner constructions [58–61], all of these cover the approaches listed here only in parts and have different foci. Therefore, this survey and its taxonomies are novel and an individual contribution of this thesis.

Survey on beaconless algorithms - Section 4 This survey systematically and comprehensively reviews beaconless algorithms. It is not limited to beaconless algorithms that directly relate to the algorithmic contributions given in this thesis, but shall provide a more general overview on this research field.

The individual algorithms are categorized according to their objectives. If applicable, the algorithms are further grouped into subcategories according to their modes of operation and/or specific objectives. A generalized mode of operation is extracted and described, whenever possible. Furthermore, tabular taxonomies are presented that allow for comparison of related approaches.

To the best of the author's knowledge, this algorithm class has not yet been surveyed as a whole and no taxonomies are available. Hence, this is a novel and individual contribution of this thesis.

Concept of $\mathcal{O}$- and Ω-reactive topology control - Chapter 5 This concept mathematically formalizes and generalizes the formerly imprecise notion of *reactive* (aka *beaconless*) *topology control* and its underlying problem statement.

Distributed message complexity is used as a means to define the classes of $\mathcal{O}$- and Ω-reactive topology control algorithms. These classes enable the distinction of reactive from conventional, beacon-based, distributed algorithms, and facilitate classification of reactive algorithms. The application to existing approaches yields a taxonomy which reveals research gaps and peculiarities of the current state of research. Furthermore, based on these formalisms, two fundamental propositions regarding the reactive computability of standard topology control structures are proven. That is, this contribution does not merely serve as a tool for classification of algorithms, but also constitutes a theoretical foundation for in-depth investigation of this algorithm classes' principal power.

Both the fundamental propositions as well as the classification scheme are used throughout this thesis to analyze and discuss the quality of reactive topology control approaches that are introduced in this work.

Euclidean spanning ratio of the partial Delaunay triangulation - Section 6.1 It is proven that the partial Delaunay triangulation (PDT) applied on a connected unit disk graph yields a connected, planar, constant stretch Euclidean spanner, whose spanning ratio is at most $\frac{1+\sqrt{5}}{4} \cdot \pi^2$. This result finally answers the formerly open question of whether PDT is a constant stretch Euclidean spanner for the unit disk graph. Some authors [59, 62–65] have stated this to be unknown, whereas others [66, 67] have even claimed that PDT is not a constant stretch spanner at all.

This result is obtained by showing that PDT is equivalent to the partial unit Delaunay triangulation (PuDel), which was recently proven to be a $(\frac{1+\sqrt{5}}{4} \cdot \pi^2)$-Euclidean spanner for the unit disk graph [62]. Part of the contribution presented here is a completed and simplified version of the proof given in [62].

Proving PDT to have a constant Euclidean spanning ratio has several implications, which are listed as part of that section's discussion.

$\mathcal{O}$-reactive local construction of planar Euclidean unit disk spanners - Section 6.2 The $\mathcal{O}$-reactive local topology control algorithm *reactivePDT* (rPDT) is introduced and proven to be correct. Executed by any node from a given unit disk graph, it computes this node's adjacency in the partial Delaunay triangulation, which is a constant stretch Euclidean spanner for the unit disk graph.

So far only reactive local algorithms for construction of planar non-spanners have been known. This contribution positively answers the open question, if planar spanners can be constructed reactively at all [14, 15, 68]. In addition, this is the first reactive topology control approach that does not require sending additional *protest messages*, which makes it a message optimal algorithm.

Guaranteed delivery reactive geographic routing on unit disk spanners - Chapter 7 The fact that the partial Delaunay triangulation (PDT) is a constant stretch Euclidean spanner for the unit disk graph leads to several related contributions regarding guaranteed delivery reactive geographic routing. First of all, it is shown that with *Rotational Sweep* [15,68,69] in *Sweep Circle* mode (RS-SC), a guaranteed delivery, reactive geographic routing protocol which uses only edges of Euclidean unit disk spanners already exists. This result is obtained by showing that RS-SC recovery paths coincide with the corresponding PDT face traversals. The second contribution is an analysis of RS recovery path properties. Examples are given where RS performs particularly badly, when looking at the hop instead of the Euclidean metric. These considerations ultimately lead to the third contribution: the guaranteed delivery, reactive geographic routing protocol *RS-Shortcut*. Routing paths produced by this algorithm use at most as many hops as those produced by RS-SC, skip arbitrarily many unnecessary hops, and use only edges of PDT. At the same time this algorithm requires at most a constant number of message transmissions per forwarding step. With respect to these theoretical guarantees, RS-Shortcut is currently the most advanced reactive geographic routing protocol.

Ω-reactive local construction of planar topological quasi unit disk spanners and reactive geographic routing thereon - Chapter 8 The Ω-reactive local topology control algorithm *ReactiveBackbone* is introduced and proven to be correct. Executed by any node from a quasi unit disk graph, satisfying that the ratio of the maximum to the minimum communication radius is bounded by $\sqrt{2}$, this algorithm computes the node's local view on a connected *routing graph* whose topological spanning ratio is bounded by a constant. This routing graph consists of two parts: a planar backbone graph whose planarity is achieved by replacing edge intersections by virtual nodes, and one edge per non-backbone node connecting it to the backbone. It can be extended by means of algorithm rPDT in order to also guarantee constant Euclidean stretch properties.

ReactiveBackbone is the first local algorithm designed for the construction of a node's local view on a quasi unit disk graph spanner and, in particular, it is the first reactive approach. All previous techniques are designed for network-wide operation or require additional assumptions. Moreover, all previous approaches are conventional, beacon-based approaches. Regarding its message-complexity the algorithm is worst-case optimal up to

constant factors and outperforms any beacon-based counterpart. The routing graph and the virtual planar backbone itself are not novel, but were introduced previously in [70, 71]. As a part of this contribution, however, it is shown that these graphs are generated by a k-local topology mapping, for $k = 8$.

Based on these results and algorithm *Virtual-Face-Traversal* [72, 73] the guaranteed delivery, reactive geographic routing algorithm *Reactive-Virtual-Face-Traversal* is introduced. It is the first reactive geographic routing algorithm which routes messages along the edges of a constant stretch Euclidean and topological spanner under the given model assumptions.

Asynchronous planarization of quasi unit disk graphs - Chapter 9 The final contribution of this thesis is a digression on planarization and guaranteed delivery geographic routing on quasi unit disk graphs in the absence of any node synchronization. The correctness of reactive algorithms heavily relies on perfect synchronization as well as on reliable and instantaneous transmissions. Therefore, such algorithms fail in completely asynchronous networks. As a first step to remedy this situation, algorithm AsyncPDT is introduced and proven to be correct. If executed by all nodes of a connected quasi unit disk graph, satisfying that the ratio of the maximum to the minimum communication radius is bounded by $\sqrt{2}$, a connected and planar overlay graph is constructed that allows for application of any guaranteed delivery geographic routing algorithm. At first, virtual edges are added. Afterwards, this supergraph is planarized by means of the partial Delaunay triangulation (PDT). The output graphs of AsyncPDT are denser than those from related approaches. Furthermore, it is shown that for the class of civilized graphs, AsyncPDT can be transformed into a local algorithm, which can be used for local topology control and, therefore, is a good starting point for reactive algorithms research in asynchronous systems.

1.4 Own Publications Used in this Thesis

The contributions presented in this thesis are mainly based on the following publications by the author. Contents and parts of these particular papers, such as verbatim quotations, ideas, explanations, descriptions, proofs, figures, and tables have been taken over without additional and explicit citation.

F. Neumann and H. Frey, "On the Spanning Ratio of Partial Delaunay Triangulation," in *Proc. of the 9th IEEE Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, LV, NV, USA, Oct. 2012, pp. 434–442. [74]

Contents: The paper proves the partial Delaunay triangulation to be a constant stretch Euclidean spanner for the unit disk graph.

Used in: Section 6.1

Remark on authorship: The paper was authored by F. Neumann and co-authored by H. Frey.

M. Benter, F. Neumann, and H. Frey, "Reactive Planar Spanner Construction in Wireless Ad Hoc and Sensor Networks," in *Proc. of the 32nd IEEE Intl. Conf. on Computer Communications (INFOCOM)*, Torino, Italy, Apr. 2013, pp. 2193–2201. [75]

Contents: The paper introduces an $\mathcal{O}$-reactive algorithm for construction of a node's adjacency in the partial Delaunay triangulation.

Used in: Section 6.2

Remark on authorship: The paper was authored by F. Neumann and co-authored by M. Benter and H. Frey. The algorithm and its proof of correctness were jointly derived by M. Benter and F. Neumann in the course of M. Benter's Master's thesis (see reference below), which was supervised by F. Neumann and H. Frey.

M. Benter, "Reactive Message-Efficient Geographic Routing Using Partial Delaunay Triangulation," *Master's thesis*, University of Paderborn, Germany, Oct. 2012.

F. Neumann and H. Frey, "Path Properties and Improvements of Sweep Circle Traversals," in *Proc. of the IEEE 9th Intl. Conf. on Mobile Ad-hoc and Sensor Networks (MSN)*, Dalian, China, Dec. 2013, pp. 101–108. [76]

Contents: The paper proves Sweep Circle traversals to use edges of constant stretch Euclidean unit disk graph spanners and introduces a superior beaconless recovery strategy.

Used in: Chapter 7

Remark on authorship: The paper was authored by F. Neumann and co-authored by H. Frey.

F. Neumann, C. Botterbusch, and H. Frey, "Probing Message Based Local Optimization of Rotational Sweep Paths," in *Ad-hoc, Mobile, and Wireless Networks: Proc. of the 13th Intl. Conf. on Ad-hoc, Mobile, and Wireless Network (ADHOC-NOW)*, ser. LNCS, S. Guo, J. Lloret, P. Manzoni, and S. Ruehrup, Eds. Benidorm, Spain: Springer International Publishing, Jun. 2014, vol. 8487, pp. 58–71. [77]

Contents: The paper introduces a probing-based approach to avoid unnecessary forwarding operations during Rotational Sweep recovery. Via mathematical analysis and simulations it is shown to reduce energy consumption of recovery operations.

Used in: Chapter 7

Remark on authorship: The paper was authored by F. Neumann and co-authored by C. Botterbusch and H. Frey. The algorithm and its simulation were jointly derived by C. Botterbusch and F. Neumann in the course of C. Botterbusch' Bachelor's thesis (see reference below), which was supervised by F. Neumann and H. Frey. The generalized mathematical analysis was joint work of H. Frey and F. Neumann.

C. Botterbusch, "Reaktive Protokolle zur Optimierung von Rotational Sweep Recovery Pfaden," *Bachelor's thesis (in German)*, University of Koblenz-Landau, Germany, Feb. 2014.

F. Neumann and H. Frey, "Partial delaunay triangulation-based asynchronous planarization of quasi unit disk graphs," in *Proc. of the 2nd Intl. Conf. on Networked Systems (NetSys)*, Cottbus, Germany, Mar. 2015, pp. 1–8. [78]

Contents: The paper introduces an asynchronous distributed algorithm for construction of the partial Delaunay triangulation over quasi unit disk graphs.

Used in: Chapter 9

Remark on authorship: The paper was authored by F. Neumann and co-authored by H. Frey.

F. Neumann and H. Frey, "Foundation of Reactive Local Topology Control, "*IEEE Commun. Lett.*, vol. 19, no. 7, pp. 1213–1216, Jul. 2015. [79]

Contents: The paper introduces a concise mathematical model of reactive topology control. This model is used to classify existing approaches and to derive fundamental impossibility results.

Used in: Chapter 5

Remark on authorship: The paper was authored by F. Neumann and co-authored by H. Frey.

F. Neumann and H. Frey, "On Demand Beaconless Planar Backbone Construction for Quasi Unit Disk Graphs," in *Proc. of the 12th Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, Dallas, TX, USA, Oct. 2015, pp. 342–351. [80]

Content: The paper presents an Ω-reactive topology control algorithm for construction of planar and constant stretch topological spanners over quasi unit disk graphs and its application to reactive geographic routing.

Used in: Chapter 8

Remark on authorship: The paper was authored by F. Neumann and co-authored by H. Frey. Parts of the contents regarding beaconless clustering were jointly derived by J. Mosen and F. Neumann in the course of J. Mosen's Bachelor's thesis (see reference below) which was supervised by F. Neumann and H. Frey.

J. Mosen, "Ein Reaktiver Algorithmus für Geografisches Clustering," *Bachelor's thesis (in German)*, University of Koblenz-Landau, Germany, May 2014. [Online]. Available: `https://kola.opus.hbz-nrw.de/frontdoor/index/index/docId/851`

Lastly, parts of the survey on beaconless algorithms in Chapter 4 are joint work of D. Vivas Estevao and F. Neumann in the course of D. Vivas Estevao's Bachelor's thesis (see reference below), which was supervised by F. Neumann and H. Frey.

D. Vivas Estevao, "Eine systematische Literaturstudie zu beaconless Algorithmen für drahtlose Ad-hoc- und Sensornetze," *Bachelor's thesis (in German)*, University of Koblenz-Landau, Germany, Oct. 2014. [Online]. Available: `https://kola.opus.hbz-nrw.de/frontdoor/index/index/docId/883`

1.5 Outline

Chapter 2 introduces fundamental concepts and definitions that are required to understand the contributions of this thesis.

Chapter 3 systematically and comprehensively surveys local and localized techniques for planarization of unit disk and quasi unit disk graphs. The part on unit disk graphs is limited to approaches which provably output connected and planar graphs. The part on quasi unit disk graphs also covers approaches that produce non-planar output graphs.

Chapter 4 systematically and comprehensively surveys beaconless algorithms. It provides a general overview on this research field. The scope of this survey is limited to local(ized) beaconless algorithms that do not make use of flooding.

Chapter 5 presents the concepts of $\mathcal{O}$- and Ω-reactive local topology control which are then used for classification of existing approaches and derivation of fundamental propositions.

Chapter 6 is dedicated to reactive local construction of Euclidean spanners under unit disk model assumptions. In Section 6.1 it is first shown that the partial Delaunay triangulation is a constant stretch Euclidean spanner for the unit disk graph. Subsequently, in Section 6.2 the $\mathcal{O}$-reactive local topology control algorithm rPDT for construction of the partial Delaunay triangulation is introduced.

Chapter 7 applies the findings from Chapter 6 to state-of-the-art beaconless recovery routing. Based thereon, new properties for existing approaches are derived and the recovery algorithm RS-Shortcut is presented.

Chapter 8 is dedicated to reactive local topology control and geographic routing under quasi unit disk model assumptions. The Ω-reactive local topology control algorithm ReactiveBackbone is presented, which constructs planar, constant stretch topological spanners. Based thereon, the guaranteed delivery reactive geographic routing protocol Reactive-Virtual-Face-Traversal is introduced.

Chapter 9 introduces the asynchronous distributed algorithm AsyncPDT for construction of planar graphs over quasi unit disk graphs.

Chapter 10 concludes this thesis. It is discussed to which extent the presented contributions match the research objectives. Common aspects and implications of the contributions are further discussed and directions for future research are given. The thesis closes with some final remarks.

Chapter 2

Fundamentals

This chapter introduces fundamental concepts and definitions that are required to understand the contributions of this thesis. In Section 2.1 the modeling of wireless ad hoc and sensor networks, by so called *unit disk* and *quasi unit disk graphs*, is introduced and discussed. The present work focuses on design and analysis of *local algorithms* under these model assumptions. This term is defined and differentiated from so called *localized algorithms* in Section 2.2. In that context, the notion of *distributed message complexity* is introduced, which is a central metric for analysis of algorithmic efficiency that is used frequently throughout this work. Section 2.3 explains the principles of *geographic routing* and in particular of *local geographic routing*. The latter can be considered a key concept for this thesis, since all contributions are strongly related to problems that arise within that context. *Topology control*, another key term, is described in Section 2.4. For the presentation of related work and the results in succeeding chapters, it is indispensable to introduce some concepts from *geometric graph theory* and *computational geometry*[1], the subjects of Section 2.5.

2.1 Modeling of Wireless Ad Hoc and Sensor Networks

There is a wide spectrum of models for wireless ad hoc and sensor networks, starting with rather simplistic connectivity models such as the *unit disk graph* (UDG) [81], to the extreme, very technical models that take interference into account, such as the *Signal-to-Interference Plus Noise* (SINR) model. Schmid and Wattenhofer survey the most common models in [82]. The two models selected for this work are introduced and discussed next.

Definition 2.1 (Unit disk graph (UDG) [82])**.** Let $V \subset \mathbb{R}^2$ be a set of nodes in the Euclidean plane and $\mathrm{R} > 0$ a fixed constant. The Euclidean graph $G = (V, E)$ is called a *unit disk graph* (UDG), if any two nodes are adjacent if and only if their Euclidean distance is at most R. That is, for any $u, v \in V$, edge uv is contained in E if and only if $\|uv\| \leq \mathrm{R}$. Typically this graph is denoted by $\mathrm{UDG}(V)$.

The main weakness of this model is that the *transmission radius* (also known as *radio coverage area* or *communication range*) is assumed to be a perfect circle, which is critical in non-ideal, obstructed environments. Moreover, the area and shape of a node's transmission range is influenced by other factors such as antenna patterns and environmental

[1] Although these contents are well-known and often considered to be well established facts, they are revisited so that this thesis is complete and self-contained.

conditions. The quasi unit disk graph (QUDG) [83, 84] model generalizes UDG and partially remedies its shortcoming by taking imperfections of the nodes' communication ranges and obstructions of the deployment area into account.

Definition 2.2 (Quasi unit disk graph (QUDG))**.** Let $V \subset \mathbb{R}^2$ be a set of nodes in the Euclidean plane and $0 < \mathrm{r} \leq \mathrm{R}$ two fixed constants, henceforth referred to as *minimum transmission radius* r and *maximum transmission radius* R. In the quasi unit disk graph over V, denoted $\mathrm{QUDG}(V)$, two nodes $u, v \in V$ are connected by a bidirectional edge if $\|uv\| \leq \mathrm{r}$, whereas they are not connected if $\|uv\| > \mathrm{R}$. In the remaining case where $\mathrm{r} < \|uv\| \leq \mathrm{R}$, the bidirectional edge uv may or may not be present. In the special case where $\mathrm{R} = \mathrm{r}$, this model is equivalent to UDG.

This definition does not further specify under which conditions two nodes $u, v \in V$ with $\mathrm{r} < \|uv\| \leq \mathrm{R}$ are (dis)connected. Several options are proposed in [82]. In this thesis, an adversarial connection of nodes is assumed, in order to challenge the universality of the algorithmic solution at hand.

By adjusting the ratio of the maximum to the minimum communication radius, this model can effectively model the presence of physical obstructions and irregularity of transmission ranges. It can also model uncertainty in geographic node positions (see [71] and the discussion in Section 10.2 for further details).

For modeling of certain environments such as urban scenarios, QUDG may still be deficient and not be able to capture every observable degeneration in real-world deployments. However, "including all these details in the network model would make it extremely complicated and scenario-dependent, hampering the derivation of meaningful and sufficiently general analytical results." [85, p.169]

Although UDG model assumptions are rather simplistic, there are still good reasons to start the algorithm design process using this model. Firstly, there are algorithms developed for UDG which also perform well in more general models [82]. Secondly, many algorithms designed for UDG can be transformed to work under the more realistic QUDG model, although with additional costs [82, 86]. Thirdly, the present work focuses on the general power of beaconless topology control, a research field for which there is currently only very limited understanding. It is a reasonable and common approach to start deriving results beginning with a rather simplistic model, and then to gradually generalize these results while relaxing model assumptions.

2.2 Local vs. Localized Algorithms and Message Complexity

Although the terms *local algorithm* and *localized algorithm* are often used synonymously, these terms describe two special cases of what is generally referred to as a *distributed algorithm*.[2] Wattenhofer [88] defines localized algorithms as follows:

Definition 2.3 (Localized algorithm [88])**.** In a *k-localized algorithm*, for some parameter k, each node is allowed to communicate at most k times with its neighbors. A node can

[2] For details on the notion of *distributed algorithms* the reader is referred to the textbook on distributed computing by Peleg [87]

decide to retard its right to communicate; for example, a node can wait to send messages until all its neighbors with larger identifiers have reached a certain state of their execution.

Such algorithms are *local* in the sense that each node communicates only with nodes in its k-hop neighborhood. However, by allowing a node to retard its next communication step until other network nodes have reached a certain state, inherently non-local information flow is introduced. For instance, this may be the case if a node u waits for state change of its neighbor v, which in turn waits for state change of its neighbor w, and so on. In the worst-case, there can be a *linear chain of causality* [88] such that only one node is active at a point in time and hence, the algorithm requires $\Theta(n)$ many rounds of communication. This example also illustrates that a node's decision (e.g., on state change) in localized algorithms inherently depends on the decisions of nodes that are arbitrarily many hops apart. Therefore, in a strict sense, approaches satisfying the above definition are not local. Wattenhofer [88] and Suomela [60] define *local algorithms* in the following, more restrictive sense.

Definition 2.4 ((Synchronous) local algorithm [88])**.** In a *k-local algorithm*, for some parameter k, any node can communicate at most k times with its neighbors, but in contrast to k-localized algorithms, nodes are not allowed to retard their right to communicate. In particular, all nodes process k synchronized phases, and a node's operations in phase i may only depend on the information received during phases 1 to $i-1$. An algorithm is said to be *local* if it is k-local for some integer constant k.

This definition of local algorithms is very narrow as it covers only those algorithms where communication proceeds in synchronized phases (rounds). In the context of wireless ad hoc and sensor networking, however, there are many local algorithms which do not fit the above definition because communication is not performed in synchronized phases. This holds in particular for beaconless algorithms. Therefore, based on a formulation given in [60], the following alternative and less restrictive definition of local algorithms, encompassing the algorithm class defined by Definition 2.4, shall be used throughout this thesis.

Definition 2.5 (Local algorithm)**.** In a *k-local algorithm*, for some parameter k, the output of a node is a function of the input of those nodes that are connected to it by at most k hops [60]. Here, the *input of a node* is task-specific and refers, e.g., to the node's unique identifier or geographic position. An algorithm is said to be *local* if it is k-local for some integer constant k.

In local algorithms, a node can gather information only about those nodes that are at most k hops apart. In particular, implicit state propagation as in the case of localized algorithms is impossible. Hence, this notion captures that such algorithms are completely free of any centralized control [89]. For further details and discussions, the reader is referred to [60, 87–89] and the references given therein.

One major metric for analyzing the efficiency of a distributed algorithm is given by its *worst-case message complexity*.

Definition 2.6 (Message complexity [87])**.** The *worst-case message complexity* of a distributed algorithm is the total number of messages sent during its execution in the worst-case over any legal input.

In order to reduce communication overhead, it is of importance to design algorithms with low message complexity.

2.3 Geographic Routing

Geographic routing (synonymously referred to as *georouting* [34] and *position-based routing* [90]) is the class of routing protocols which base routing decisions on geographic node positions. This presupposes that network nodes are provided with their geographic positions or are able to determine these by means of some positioning system such as the *Global Positioning System* (GPS). Furthermore, geographic routing requires that the position of a packet's destination is either known in advance, or has been determined prior to the actual routing process, using so called *location services*. For a general overview on geographic routing see [90, 91] and for surveys on geographic routing protocols and location services see [92, 93].

Local geographic routing algorithms (often also referred to as *localized geographic routing algorithms*) are distributed geographic routing algorithms that base their forwarding decisions on partial knowledge of the network graph, as well as partial knowledge of the routing path of the packet that has to be forwarded. Specifically, in this algorithm class a forwarding node v is only provided with its own position, the position of the packet's destination, positions of its k-hop neighbors $\mathrm{N}_k(v)$ in the network graph (typically $k = 1$), and a constant amount of information on the previous routing path of the packet (e.g., a constant number of node positions visited by the packet). That is, local geographic routing algorithms are (nearly) stateless routing algorithms. Generally, a routing algorithm is said to be *stateless* if it is local, only a constant number of node addresses are stored in the packet header, and nodes do not have to memorize past routing decisions for processing a novel routing request [90].

A (local) geographic routing algorithm is said to be *loop-free*, if it does not cause forwarding loops. Moreover, it is said to *guarantee delivery*, if the packet is successfully routed from its source to its destination using only a finite number of routing steps, provided that source and destination are connected, and assuming an ideal, collision-free access scheme. Geographic routing algorithms differ in their path strategies. In *single path* algorithms, only one copy of the packet is in the network at any time. The other extreme are *flooding*-based approaches, where the packet is flooded through parts or even the entire network. *Multipath* strategies route several copies of a packet along different paths, which are typically disjoint [90].

(Local) geographic routing is not dedicated to *unicast routing*, i.e., routing of a packet from one source to one destination. As discussed later in Section 4.4, there are also geographic routing algorithms for *multicast routing*. The latter refers to sending of a packet from one source to multiple destinations. Geographic multicast routing presupposes that the *multicast group*, i.e., positions of all intended receivers, is known in advance.

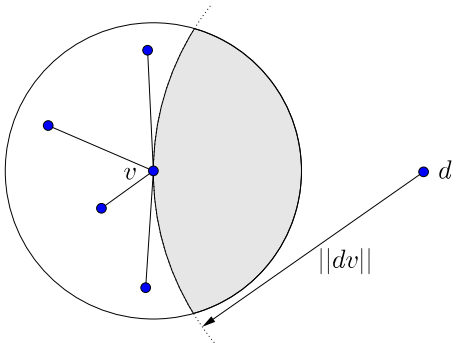


Figure 2.1: Example of a local minimum situation. Forwarder v has no neighbor in its transmission range (circle) providing positive progress towards the destination node d. The shaded area represents the *positive progress area* (PPA) of v, the area where nodes are closer to the destination than v itself.

Local geographic routing algorithms typically consist of two components: *Greedy routing* and *Recovery routing*.

2.3.1 Greedy Routing

Greedy routing is the principle of positively advancing the packet in each routing step as much as possible with respect to a specific metric. This idea was first introduced by Finn [94]. The node currently holding the packet, called *forwarder* or *forwarding node*, selects the neighbor for forwarding, referred to as *next-hop*, which minimizes the Euclidean distance to the destination.

Greedy routing fails in the case of a *local minimum situation*[3](also called *local optimum* [40] or *void* [96]). This refers to scenarios where the forwarder is neither able to forward the message directly to the destination, nor does any of its neighboring nodes provide positive progress towards the destination. An example of a local minimum situation is given in Figure 2.1. Formally it is defined as follows:

Definition 2.7 (Local minimum)**.** Given a network graph $G = (V, E)$ embedded in $\mathbb{R}^2$, a node $v \in V$ is said to be a *local minimum* w.r.t. some destination node $d \in V$, if $d \notin \mathrm{N}_1(v)$ and $\forall\, u \in \mathrm{N}_1(v) : \|ud\| \geq \|vd\|$.

A local minimum situation is given, if and only if the *positive progress area* (PPA), which is defined as the intersection of the transmission radius of v and the circle centered at d with radius $\|dv\|$, of a forwarder v w.r.t. destination d is empty of other network nodes. In Figure 2.1, the PPA is represented by the shaded area.

[3] This does not hold for Greedy routing algorithms like *geographical distance routing* (GEDIR) [95], which allow a message to travel backwards for one hop. However, this can be considered a simple recovery heuristic and therefore, such schemes are not "pure" Greedy routing strategies.

2.3.2 Recovery Routing

Let v be a local minimum node w.r.t. destination d. The purpose of *Recovery routing* is to route the packet from v to a network node w, such that either w or a neighboring node of w is closer to the destination than v. Formally, the underlying problem, called *Greedy recovery problem* or simply *recovery problem*, can be defined as follows.

Definition 2.8 (Greedy recovery problem)**.** Let $G = (V, E)$ be a connected network graph embedded in $\mathbb{R}^2$ s.t. $v \in V$ is a local minimum w.r.t. $d \in V$. The *Greedy recovery problem* refers to the problem of constructing a *recovery path* $\mathcal{P} = \langle v = u_0, ..., u_k = w \rangle$ in G, s.t. either $\|wd\| < \|vd\|$, or $\exists\, w' \in \mathrm{N}_1(w)$ with $\|w'd\| < \|vd\|$, and where no edge on path $\mathcal{P}$ is visited twice in the same direction (i.e, the recovery path is free of loops).

The most noted recovery strategy for local geographic routing algorithms is *FACE* routing, which was initially introduced by Bose et al. [57]. FACE routing is a stand-alone, guaranteed delivery, local geographic routing algorithm that works independently of Greedy routing. It can also be combined nicely with Greedy routing to *Greedy-Face-Greedy* (GFG) [36], which is introduced right after the explanation of FACE routing. The latter requires the notion of *planar graphs*.

Definition 2.9 (Planar graph)**.** Traditionally, a graph is said to be *embeddable in the plane*, or *planar*, if it can be drawn in the plane so that its edges intersect only in their endpoints. Such a drawing is called a *planar embedding* of the graph [97].

In the context of wireless ad hoc networks, including this thesis, the term *planar graph* is used in the following and more specific sense: Given a graph whose nodes are embedded in the Euclidean plane and whose edges are represented by straight line segments connecting these nodes, a graph is said to be *planar* if the edges intersect only in their endpoints.

A planar network graph $G = (V, E)$ partitions the Euclidean plane into several *inner faces* and one *outer face* (see Figure 2.2).

Starting from some node s, FACE routing forwards the packet with destination d along the interiors of one or several adjacent faces which are intersected by the straight line connecting s and d, e.g., faces F_1, F_2, F_3, F_4 in Figure 2.2. Each face interior is traversed using the well-known *right-hand-rule* [*left-hand-rule*], such that the packet is forwarded clockwise [counterclockwise] from the previous edge. Whenever the packet reaches an edge which intersects the *sd*-line, this edge is skipped and the next adjacent face intersecting the *sd*-line is traversed in the same way. The algorithm is executed until either the destination is reached, or the very first edge is used for a second time in the same direction, in which case it can be concluded that d is not contained in the same network component and hence, is unreachable.

Frey and Stojmenović [98] show FACE routing to be loop-free and to guarantee delivery if it is applied on static connected, planar graphs embedded in the Euclidean plane.[4]

FACE routing can be combined with Greedy routing as follows: The packet is forwarded using Greedy routing until a local minimum node v is reached. Then, the packet is switched

[4] In addition, [98] surveys many FACE routing variants and (dis)proves their delivery guarantees.

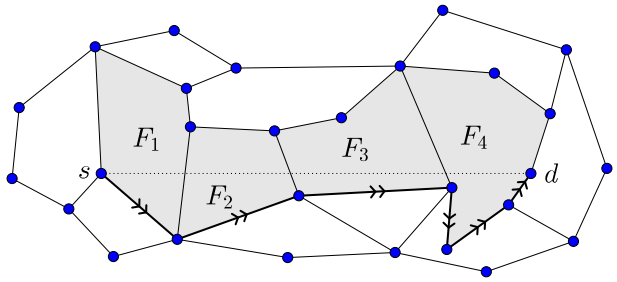


Figure 2.2: Illustration of the FACE routing principle using the right-hand-rule. Starting from node s, the packet is forwarded along the faces F_1, F_2, F_3, F_4 using the highlighted edges until it reaches destination d.

from *Greedy mode* into *Recovery mode* and FACE routing is used until a node closer to the destination than v has been reached. At this node, the packet is then switched back to *Greedy mode* and forwarded using Greedy routing again. When switched to Recovery mode, the packet header is extended by the *recovery distance*, the Euclidean distance between the local minimum node and the destination, as well as by the *first edge used in Recovery mode*. Given this information, each node receiving the packet in Recovery mode can determine if it can switch back to Greedy mode. This is the case if it is closer to the destination than the recovery distance, or if the packet is undeliverable. The latter applies if the next FACE routing step is similar to the very first one made by the local minimum node.

This combination of Greedy and FACE routing is known as *Greedy-Face-Greedy* (GFG). It was first introduced by Bose et al. in [36] and was later extended by Karp and Kung [37] to the well-known *greedy perimeter stateless routing* protocol GPSR by including the IEEE 802.11 medium access control scheme. In general, routing paths produced by GFG are shorter compared to those using only FACE routing [36].

For a survey on other recovery techniques, which are not necessarily local/localized, and do not necessarily provide delivery guarantee, see the survey on void handling techniques by Chen and Varshney [96].

2.4 Topology Control

In dense wireless network graphs, where density is measured e.g. by means of average number of neighbors of a node, typical wireless networking problems arise [3].

- Many nodes interfere with each other.
- There are lots of possible routes available.
- Nodes might need needlessly large transmission power to exchange messages with distant nodes, which reduces the reuse of available bandwidth.
- In mobile networks, routing protocols may have to recompute routing tables even if only small changes have happened.

Generally speaking[5], *topology control* refers to the transformation of a given network graph $G = (V, E)$ into another graph $G' = (V', E')$ that obeys certain desirable properties and thereby alleviates some of the aforementioned problems [3].

For the reasons given in detail in the introduction, the present work is particularly interested in *local topology control algorithms* for *geometric input graphs* $G = (V, E)$, i.e., graphs where $V \subset \mathbb{R}^2$ and E is a set of straight lines embedded in the Euclidean plane, which produce output topologies $G' = (V', E')$ satisfying the following properties:

Connectivity If G is a connected graph, then G' is a connected graph.

Planarity G' is *planar* (see Definition 2.9).

Low stretch The shortest path distance between any two nodes in G is stretched by at most a small (preferably constant) factor in G'. In this work, shortest path distances are measured in terms of number of hops, or Euclidean shortest path length (see Section 2.5 for further details).

Mainly two types of output topologies are considered in this work: Given an input graph $G = (V, E)$, then G' is either a *subgraph* of G with $G' = (V, E' \subseteq E)$, or G' is a *backbone graph* for G. Generally, a backbone of a network graph $G = (V, E)$ is a *connected dominating set* (CDS) for G. A subset $S \subseteq V$ is called dominating set if every node either belongs to S, or is adjacent to a node from S in G. Furthermore, a dominating set is *connected* if any two dominating set nodes in S are connected by a path in G that consists only of nodes from S.

2.5 Selected Terms in Geometric Graph Theory and Computational Geometry

In the following let $V \subset \mathbb{R}^2$ always be a finite and distinct set of nodes of size $|V| = n$.

2.5.1 Trees, Paths, Dilation, and Spanning Ratio

Definition 2.10 (Euclidean Minimum Spanning Tree (EMST))**.** The *Euclidean Minimum Spanning Tree* (EMST) over V, denoted EMST(V), is the minimum weight spanning tree over the complete graph whose node set is V and whose edges are weighted by the function $\omega(u, v) = \|uv\|$, $\forall\, u, v \in V$.

[5] Some authors define topology control in a much narrower sense. For instance, Santi [6] defines topology control as "the art of coordinating nodes' decisions regarding their transmitting ranges, in order to generate a network with the desired properties (e.g. connectivity) while reducing node energy consumption and/or increasing network capacity [6, p.30]." This definition obviously excludes all those algorithms that do not use adaption of transmitting ranges, but use other mechanisms and methods for generation of a desired output topology. In the remainder of this work, the notion of topology control is used in the more general sense defined above.

Definition 2.11 (Euclidean/Hop shortest path)**.** Let $G = (V, E)$ be a geometric graph. The *Euclidean length of a path* $\Pi^G(v_0, v_k) = \langle v_0, v_1, ..., v_k \rangle$ of a sequence of neighboring nodes $v_i \in V$ is the sum of the weights of its constituent edges:

$$||\Pi^G(v_0, v_k)|| = \sum_{i=1}^{k} \|v_{i-1}v_i\| \,.$$

The *Euclidean shortest path* between any two nodes $u, v \in V$, denoted $\Pi^G_{min}(u, v)$, then refers to the (not necessarily unique) path in G having smallest Euclidean length among all paths connecting u and v in G.

The *hop length of a path* $\Pi^G(v_0, v_k)$ is simply the number of its constituent edges:

$$\delta(\Pi^G(v_0, v_k)) = \sum_{i=1}^{k} 1 \,.$$

The *hop shortest path* between any two nodes $u, v \in V$, denoted $\mathrm{H}^G_{min}(u, v)$, then refers to the (not necessarily unique) path in G having smallest hop length among all paths connecting u and v in G.

Definition 2.12 (Dilation)**.** Let $G = (V, E)$ be a geometric graph. The *dilation* $\Delta(u, v)$ of two nodes $u, v \in V$ refers to the ratio of the Euclidean shortest path between u and v in G to their Euclidean distance $\|uv\|$. Formally,

$$\Delta(u, v) = \frac{||\Pi^G_{min}(u, v)||}{\|uv\|} \,.$$

The *dilation* of graph G, denoted $\Delta(G)$, is defined as the maximum dilation between any two nodes $u, v \in V$, formally

$$\Delta(G) = \max_{u,v \in V} \{\Delta(u, v)\} \,.$$

Definition 2.13 (Euclidean t-spanner/Euclidean stretch factor/Euclidean spanning ratio)**.** Let $G = (V, E)$ be a geometric graph and $H = (V, E' \subseteq E)$ a subgraph of G. It is synonymously said that

- H is a *Euclidean t-spanner* of G,
- H has *Euclidean stretch factor t* w.r.t. G, and
- H has *Euclidean spanning ratio t* w.r.t. G,

if it holds that

$$\max_{u,v \in V} \left\{ \frac{||\Pi^H_{min}(u, v)||}{||\Pi^G_{min}(u, v)||} \right\} \leq t \,.$$

If t is a constant, H is often referred to as a *constant stretch Euclidean spanner*, or even as a *Euclidean spanner* if it is clear from the context. Synonymously it is said that H has a constant Euclidean spanning ratio.

In analogy to the notion of Euclidean spanners, there is another notion called *Euclidean weak spanner*. Unlike Euclidean spanners, weak spanners do not ensure that shortest Euclidean paths are stretched only by constant factors, but rather ensure that there always exist paths which do not leave a specific geographic area. Formally this notion is defined as follows.

Definition 2.14 (Euclidean weak t-spanner)**.** Let $G = (V, E)$ be a geometric graph and $H = (V, E' \subseteq E)$ be a subgraph of G. Subgraph H is referred to as *Euclidean weak t-spanner*, if for any two nodes $u, v \in V$ there exists a path $\Pi^H(u, v)$ connecting u and v such that $\Pi^H(u, v)$ is covered entirely by the disk of radius $t \cdot \|uv\|$, centered at the midpoint of uv.

Definition 2.15 (Hop t-spanner/Hop stretch factor/Hop spanning ratio)**.** Let $G = (V, E)$ be a graph and $H = (V, E' \subseteq E)$ a subgraph of G. It is synonymously said that

- H is a *hop/topological t-spanner* of G,
- H has *hop/topological stretch factor t* w.r.t. G, and
- H has *hop/topological spanning ratio t* w.r.t. G,

if it holds that

$$\max_{u,v \in V} \left\{ \frac{\delta(\mathrm{H}^H_{min}(u, v))}{\delta(\mathrm{H}^G_{min}(u, v))} \right\} \leq t \,.$$

If t is a constant, H is often referred to as *constant stretch hop/topological spanner*, or even as *hop/topological spanner* if it is clear from the context. Synonymously it is said that H has a constant hop/topological spanning ratio.

2.5.2 Voronoi Diagram, Delaunay Triangulation, and Proximity Graphs

Given a line segment $\overline{uv}$ between two nodes $u, v \in V$, the *perpendicular bisector* of u and v is given by

$$\mathrm{B}(u, v) = \{x \in \mathbb{R}^2 : \|ux\| = \|vx\|\} \,.$$

$\mathrm{B}(u, v)$ divides the Euclidean plane into the *closed half-plane*

$$\mathrm{H}(u, v) = \{x \in \mathbb{R}^2 : \|ux\| \leq \|vx\|\}$$

containing node u and the closed half-plane $\mathrm{H}(v, u)$ containing node v. While considering closed half-planes, note that $\mathrm{H}(u, v) \cap \mathrm{H}(v, u) = \mathrm{B}(u, v)$ holds.

Definition 2.16 (Voronoi diagram [99])**.** Let $u \in V$. The region

$$\mathrm{VR}_V(u) = \bigcap_{v \in V \setminus \{u\}} H(u, v)$$

is called the *(ordinary) Voronoi region* of node u w.r.t. the set V. Furthermore, set

$$\mathrm{VD}(V) = \{\mathrm{VR}_V(u_1), ..., \mathrm{VR}_V(u_n)\} \,,$$

where $\{u_1, ..., u_n\} = V$, is called the *(planar ordinary) Voronoi diagram*, generated by V (see Figure 2.3).

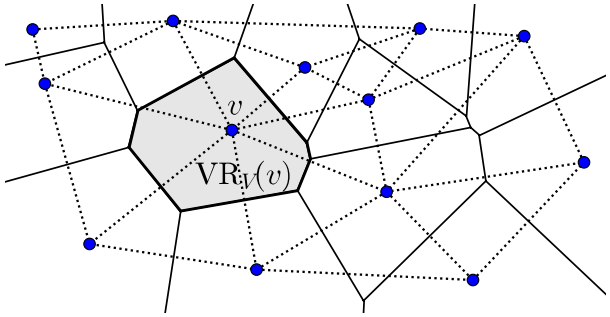


Figure 2.3: Example of a Voronoi diagram $\mathrm{VD}(V)$ (solid lines), its corresponding Delaunay triangulation $\mathrm{Del}(V)$ (dashed lines), and the Voronoi region of node $v \in V$ w.r.t. V, $\mathrm{VR}_V(v)$, which is given by the shaded area including the bold printed boundary lines.

The common boundary of two or more Voronoi regions is referred to as *Voronoi edge*, which may be a line segment, a half-line, or an infinite straight line. The endpoints of Voronoi edges are called *Voronoi vertices*. A Voronoi edge may be degenerated into a single point if it is contained on the boundary of at least four Voronoi regions. In this particular case, the notions of Voronoi edge and Voronoi vertex coincide. When there exists at least one Voronoi vertex at which four or more Voronoi edges meet in the Voronoi diagram $\mathrm{VD}(V)$, it is said to be *degenerated*, otherwise it is said to be *non-degenerated* [99]. A Voronoi diagram is always degenerated if four or more nodes in V are cocircular. As degenerated cases make proofs involving Voronoi-related concepts lengthy without adding new insights, this thesis typically adopts the following common assumption.

Assumption 2.1 (Non-cocircularity [99])**.** Every Voronoi vertex in a Voronoi diagram $\mathrm{VD}(V)$ has exactly three Voronoi edges. I.e., no four points in V are cocircular.

Definition 2.17 (Delaunay triangulation [100])**.** The *Delaunay triangulation* over set V, denoted by $\mathrm{Del}(V)$, is an undirected graph connecting each pair of nodes from V, whose Voronoi regions share a Voronoi edge in the Voronoi diagram $\mathrm{VD}(V)$ (see Figure 2.3).

Equivalently, $\mathrm{Del}(V)$ can be defined as the set of line segments $\overline{uv}$, $\forall\, u, v \in V$, for which it holds that there exists a circle that has u and v on its boundary and does not contain any node from $V \setminus \{u, v\}$ in its interior (including the boundary).

Edges in $\mathrm{Del}(V)$ are referred to as *Delaunay edges*.

In order for a Delaunay triangulation to be uniquely defined, the following assumption is required.

Assumption 2.2 (Non-collinearity [99])**.** No three points in V are collinear.

In the remainder of this thesis it is generally assumed that node sets are neither cocircular, nor collinear. While $3 \leq |V|$, V is not collinear, and V is not cocircular, a Voronoi edge in $\mathrm{VD}(V)$ can neither be an infinite straight line, nor can it degenerate into a single point [99, Section 2.3]. Moreover, under these assumptions $\mathrm{Del}(V)$ can also be

defined as the set of all triangles $\triangle(u, v, w)$ in V, $\forall\, u, v, w \in V$, for which it holds that its circumcircle (including the boundary) does not contain any other node from $V \setminus \{u, v, w\}$. In particular, under these assumptions this definition and those given in Definition 2.17 are equivalent [99, Section 2.4].

Dobkin et al. [101] proved that $\mathrm{Del}(V)$ has dilation of at most $(1 + \sqrt{5}/2)\pi \approx 5.08$. Keil and Gudwin [102] improved this upper bound to $2\pi/(3\cos(\pi/6)) \approx 2.42$. Recently, Xia [103] proved that the dilation is actually less than 1.998.

It is well-known that the Delaunay triangulation cannot be computed by a local algorithm, since the verification if an edge between two nodes belongs to it may require non-local (i.e., network wide) communication. Following the same argument, this also holds for the intersection of $\mathrm{UDG}(V)$ with $\mathrm{Del}(V)$, which is known as *unit Delaunay triangulation*, which is defined next.

Definition 2.18 (Unit Delaunay triangulation)**.** Given a $\mathrm{UDG}(V)$ with unit transmission radius R and the Delaunay triangulation $\mathrm{Del}(V)$, the *unit Delaunay triangulation* $\mathrm{UDel}(V)$ consists of all edges in $\mathrm{Del}(V)$ of length at most R, i.e.,

$$\mathrm{UDel}(V) = \{uv \in \mathrm{Del}(V) : \|uv\| \leq \mathrm{R}\}\,.$$

Edges in $\mathrm{UDel}(V)$ are called *unit Delaunay edges*.

Li et al. prove in [104] that $\mathrm{UDel}(V)$ is a constant stretch Euclidean spanner for the underlying unit disk graph whose spanning ratio is at most $((1 + \sqrt{5})/2)\pi \approx 5.08$.[6]

Generally, graphs that are defined on the basis of proximity relations between nodes are referred to as *proximity graphs* [99, p.102]. Aside from Delaunay-based proximity graphs, the *Gabriel graph* (GG) [105] as well as the *relative neighborhood graph* (RNG) [106] are the most prominent representatives of this graph family due to their numerous applications in the context of geographic routing (see e.g., [36, 37]).

Definition 2.19 (Relative neighborhood graph (RNG))**.** Any two nodes $u, v \in V$ are connected by an edge in the *relative neighborhood graph* [106] over V, denoted $\mathrm{RNG}(V)$, if and only if the proximity region defined by the intersection area $\mathrm{C}_{\|uv\|}(u) \cap \mathrm{C}_{\|uv\|}(v)$ (excluding the boundaries) does not contain any node from $V \setminus \{u, v\}$. See Figure 2.4a for an illustration. The RNG proximity region is also referred to as *RNG-lune*.

Definition 2.20 (Gabriel graph (GG))**.** Any two nodes $u, v \in V$ are connected by an edge in the *Gabriel graph* [105] over V, denoted $\mathrm{GG}(V)$, if and only if the proximity region defined by the closed disk $\mathrm{D}(u, v)$ does not contain any node from $V \setminus \{u, v\}$. See Figure 2.4b for an illustration. The GG proximity region is also referred to as *Gabriel circle* or *Gabriel disk*.

Recently, an intermediate proximity graph of GG and RNG, called *circlunar neighborhood graph* (CNG) [18, 107], has been proposed. It is defined as follows.

[6] Li et al. [104] also claim a stronger result, namely, a Euclidean spanning ratio of $((4\sqrt{3})/9)\pi \approx 2.42$, but do not give a complete proof. For this reason in later chapters only the weaker bound is used.

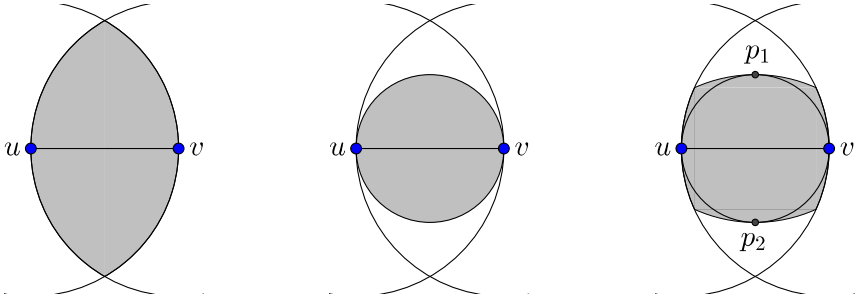


Figure 2.4: Illustrations of proximity regions and relations of proximity regions of the (a) relative neighborhood graph (RNG), (b) Gabriel graph (GG), and (c) circlunar neighborhood graph (CNG).

Definition 2.21 (Circlunar neighborhood graph (CNG))**.** Any two nodes $u, v \in V$ are connected by an edge in the *circlunar neighborhood graph* (CNG) [18,107] over V, denoted $\mathrm{CNG}(V)$, if and only if the proximity region defined by the intersection area of the circles (excluding their boundaries)

$$\bigcap_{x \in \{u,v,p_1,p_2\}} \mathrm{C}_{\|uv\|}(x)\,,$$

where p_1 and p_2 are the points of intersection of the perpendicular bisector of uv with $\mathrm{D}(u,v)$, is free of nodes from $V \setminus \{u,v\}$. See Figure 2.4c for an illustration.

The relation of the proximity regions of RNG, GG, and CNG is illustrated in Figure 2.4c. The proximity region of RNG encompasses the proximity region of CNG, which in turn encompasses the proximity region of GG. Therefore, they satisfy the following subgraph relations:

$$\mathrm{EMST}(V) \subseteq \mathrm{RNG}(V) \overset{\text{(i)}}{\subseteq} \mathrm{CNG}(V) \overset{\text{(ii)}}{\subseteq} \mathrm{GG}(V) \subseteq \mathrm{Del}(V)\,. \tag{2.1}$$

Relations (i) and (ii) in Equation 2.1 are proven in [18,107]. The remaining two relations are well established facts, see e.g., [108, p.255] and [99, Property D22]. Equation 2.1 implies in particular that RNG, CNG, and GG are connected graphs, if applied onto a finite node set V, for they contain the $\mathrm{EMST}(V)$ which is a connected graph. Moreover, since GG is planar [36, Theorem 3], this relation implies that CNG, RNG, and EMST are also planar graphs.

Bose et al. prove in [109] that RNG and GG have a dilation of $\Theta(n)$ and $\Theta(\sqrt{n})$, respectively. Rührup et al. state in [18] that the example used in [109] for proving the dilation of RNG (the so-called "RNG-tower"), can also be applied for showing that CNG has dilation of $\Theta(n)$. Moreover, the tower-constructions used in all of the these proofs imply immediately that RNG, GG, and CNG have a hop spanning ratio of $\Theta(n)$.

The aforementioned proximity graphs can also be defined over unit disk graphs. This is covered in Section 3.1 as part of the unit disk planarization techniques.

Chapter 3

Survey on Local and Localized Techniques for Graph Planarization

In this chapter a comprehensive survey on local and localized techniques for planarization of unit disk and quasi unit disk graphs is given.

Due to the quantity of publications related to this field of research and the focus of the present work, the presentation of approaches that rely on unit disk graph model assumptions in Section 3.1 is limited to those which provably output connected and planar graphs if applied onto connected input graphs. The lesser number of approaches that are designed for operation on quasi unit disk graphs allows in Section 3.2 a less restrictive choice of related work to be presented. Here, all approaches are considered that aim for construction of planar and almost planar construction of sub-, overlay, and backbone graphs.

In both sections, at first a detailed presentation of all approaches is given in a classified ordering, followed by a taxonomy that lists and summarizes the presented work.

3.1 Algorithms for Unit Disk Graphs

In the following let $\mathrm{UDG}(V)$ always denote the unit disk graph over a finite and distinct node set $V \subset \mathbb{R}^2$. Unless stated otherwise, it is assumed that $\mathrm{UDG}(V)$ is connected. As discussed previously, Delaunay- and Voronoi-based planarization techniques typically presuppose that node set V is neither collinear, nor cocircular (see Assumptions 2.1 & 2.2 in Section 2.3).

Proximity Graphs and Their Extensions

The proximity graphs RNG, GG, and CNG have previously been defined in Section 2.5.2 over the complete Euclidean graph.

The intersection of $\mathrm{RNG}(V)$, $\mathrm{GG}(V)$, and $\mathrm{CNG}(V)$ with a connected unit disk graph $\mathrm{UDG}(V)$, denoted $\mathrm{RNG}(V) \cap \mathrm{UDG}(V)$, $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$, and $\mathrm{CNG}(V) \cap \mathrm{UDG}(V)$, respectively, are connected and planar subgraphs of $\mathrm{UDG}(V)$. For proofs, see [37, Section 2.3], [36, Theorem 3], and [18, Theorem 6], respectively.

Moreover, it is easy to verify, that a node v can locally compute its one-hop neighborhood in these subgraphs by merely considering the positions of its one hop neighbors $\mathrm{N}_1(v)$ in the underlying unit disk graph.

$\mathrm{RNG}(V) \cap \mathrm{UDG}(V)$ and $\mathrm{CNG}(V) \cap \mathrm{UDG}(V)$ both have a Euclidean spanning ratio of $\Theta(n)$, whereas $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ has a Euclidean spanning ratio of $\Theta(\sqrt{n})$. The upper bounds follow from the proofs given in [18, 109] in combination with [110, Theorem 1.4]. The lower bounds can be obtained by appropriate scaling of the tower-constructions in [109]. From the latter it also follows that all of these these graphs have topological spanning ratios of $\Theta(n)$.

Based upon the observation that Gabriel graph planarization is too conservative, in the sense that some links are removed although they are actually intersection-free, Boone et al. [111] propose a simple extension of the Gabriel graph planarization rule which avoids removal of such edges. This extension is called the *Morelia test* and produces a unit disk subgraph, henceforth called *Morelia graph* (MG). A unit disk graph edge uv is always removed if both half-circles of the Gabriel disk $\mathrm{D}(u, v)$ w.r.t. uv contain other nodes from V. If only one of these half-circles contains a node w, then both u and v check, possibly by means of additional neighborhood information $\mathrm{N}_1(w)$, if there exists an edge with at least one endpoint in $\mathrm{N}_1(u) \cup \mathrm{N}_1(v)$ and one endpoint in $\mathrm{N}_1(u) \cup \mathrm{N}_1(v) \cup \mathrm{N}_1(w)$ which intersects uv. If that is the case, then uv is removed, otherwise it is maintained. The Morelia test is a 2-local algorithm and hence any node can compute its adjacency in MG using at most 2-hop neighborhood information. Because MG is a supergraph of GG, its Euclidean spanning ratio is at most as large as that of GG.

Li et al. [112] propose the *modified relative neighborhood graph* (RNG'), whose node degree is at most six. It is a subgraph of RNG and a supergraph of EMST. Hence, its Euclidean spanning ratio is $\Omega(n)$ and its topological spanning ratio is $\Theta(n)$. A UDG edge uv is maintained in RNG' if the RNG-lune w.r.t. u and v is empty of other nodes, and if there is no $w \in V \setminus \{u, v\}$ located on the boundary of the lune with

i) $ID(w) < ID(v)$ and $\|wv\| < \|uv\|$, and

ii) $ID(w) < ID(u)$ and $\|wu\| < \|uv\|$, and

iii) $ID(w) < ID(u)$, $ID(w) < ID(v)$, and $\|wu\| = \|uv\|$.

$ID(x)$ denotes the unique node identifier of a node $x \in V$. A node's adjacency in RNG' can be computed by an 1-local algorithm.

Based on the graph RNG', Li et al. [112] further propose the *light weight relative neighborhood graph* (LRNG). Its 2-local construction is as follows. Each node computes its adjacency in RNG' and exchanges this information with all 1-hop neighbors. For each RNG' edge xy received by neighbor x, node u removes each of its incident RNG' edges uv for which it holds that it is the longest of all edges in $\{uv, xy, ux, vy\}$. LRNG is symmetric, contains the EMST as a subgraph, and is itself a subgraph of RNG'. Hence, it is connected and it inherits the maximum node degree, planarity, as well as the lower bound spanning ratios from RNG'.

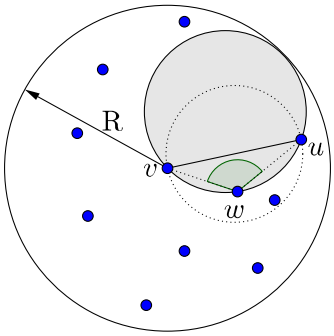


Figure 3.1: Illustration of the two PDT criteria: Although it is no GG edge, $uv \in \mathrm{UDG}(V)$ is contained in $\mathrm{PDT}(V)$ since the circle (shaded) around u, v, and the angle maximizing node w is empty of other nodes and fully contained by v's unit disk of radius R. The dotted circle represents $\mathrm{D}(u, v)$.

Voronoi- and Delaunay-based Proximity Graphs and Planarization Techniques

Li and Stojmenović [49, 50] introduce another proximity graph, called *Partial Delaunay Triangulation* (PDT). It plays a central role in this thesis. PDT is a subgraph of the unit Delaunay triangulation. It is much denser than the corresponding Gabriel graph and at the same time a node can compute its adjacency in this graph by merely inspecting its 1-hop neighborhood.

Definition 3.1 (Partial Delaunay triangulation (PDT))**.** Consider any edge $uv \in \mathrm{UDG}(V)$ and let $w \in V$, $w \neq u, v$, be the *angle maximizing node w.r.t.* uv, i.e., $\forall\, w, x \in V \setminus \{u, v\} : \measuredangle uwv \geq \measuredangle uxv$. Edge uv is contained in the *partial Delaunay triangulation* (PDT) [49, 50] over V, denoted $\mathrm{PDT}(V)$, if and only if either

1. $\measuredangle uwv < \pi/2$, (or equivalently: $uv \in \mathrm{GG}(V)$), or
2. $\sin(\measuredangle uwv) \geq \frac{\|uv\|}{\mathrm{R}}$ and $\mathrm{C}(u, v, w) \cap V \setminus \{u, v, w\} = \emptyset$.[1]

Note that PDT is GG plus those edges maintained by the second criterion. These are edges for which the Delaunay circle property can be verified based on 1-hop neighborhood information only. It ensures that the circle $\mathrm{C}(u, v, w)$ has a diameter of at most R and hence, if there would be another node $x \neq u, v, w$ located in $\mathrm{C}(u, v, w)$, then x would be a neighboring node of u, v, and w, according to the UDG model (see Figure 3.1). PDT is

[1] The angle condition formulated here is slightly weaker compared to the original definition in [49, 50], where it is required that $\sin(\measuredangle uwv) > \|uv\|/\mathrm{R}$. Hence, the definition given here produces supergraphs of those obtained by the original definition. However, these supergraphs are still connected given that the underlying UDG is connected because they contain GG. Moreover, these supergraphs are still planar, since they are subgraphs of the corresponding unit Delaunay triangulations, which are planar. Finally, a node can still decide its adjacency in this graph given its 1-hop neighborhood information. Thus, this modification of the definition of PDT does not alter the idea underlying PDT in the first place.

planar and connected if applied onto a connected unit disk graph. It is symmetric, too: if v decides to keep an edge vu, then u will decide symmetrically. In Section 6.1 it will be proven that PDT is a constant stretch Euclidean spanner of UDG, with a spanning ratio of at most $((1+\sqrt{5})/4)\pi^2 \approx 7.98$. This proof is based on a proof of equivalence to another graph construction, called *partial unit Delaunay triangulation*, which is introduced shortly.

Besides this one-hop variant of PDT, Li. et al. [49,50] introduce also a two-hop variant, called PDT_2.

Definition 3.2 (PDT_2)**.** Consider any edge $uv \in \mathrm{UDG}(V)$ and let $w \in V$, $w \neq u, v$, be the *angle maximizing node w.r.t.* uv, i.e., $\forall\, w, x \in V \setminus \{u, v\} : \measuredangle uwv \geq \measuredangle uxv$. Edge uv is contained in the *two-hop partial Delaunay triangulation* (PDT_2) [49,50] over V, denoted $\mathrm{PDT}_2(V)$, if and only if either

1. $\measuredangle uwv < \pi/2$, (or equivalently: $uv \in \mathrm{GG}(V)$), or
2. $\cos(\frac{\measuredangle uwv}{2}) \geq \frac{\|uv\|}{2\mathrm{R}}$ and $\mathrm{C}(u, v, w) \cap ((\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u, v, w\}) = \emptyset$.

PDT_2 is a connected and planar supergraph of PDT [113] and therefore it is also a constant stretch Euclidean spanner as shown in Section 6.1. Its construction requires the nodes to know their two-hop neighborhoods.

For both PDT and PDT_2 it is easy to construct worst-case examples which show that the topological spanning ratios of these graphs are $\Omega(n)$. Hence, their topological spanning ratios are $\Theta(n)$.

The definition of the *partial unit Delaunay triangulation* (PuDel), introduced by Xu et al. in [62], requires the following two auxiliary definitions.

Definition 3.3 (Directed local Delaunay edge)**.** Let $uv \in \mathrm{UDG}(V)$. Edge uv is called *directed local Delaunay edge*, denoted by $\overrightarrow{uv}$, if edge uv is contained in $\mathrm{Del}(\mathrm{N}_1(u))$.

Definition 3.4 (Locally detectable)**.** Consider any directed local Delaunay edge $\overrightarrow{uv}$. It is said to be *locally detectable*, if and only if there exists $p \in \mathbb{R}^2$ such that

1. $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$, and
2. $\|up\| \leq \mathrm{R}/2$.

Such a point p is also referred to as *witness for the local detectability* of a directed local Delaunay edge. For an illustration see Figure 3.2.

Definition 3.5 (Partial unit Delaunay triangulation (PuDel))**.** The *partial unit Delaunay graph* over a unit disk graph $\mathrm{UDG}(V)$, denoted by $\mathrm{PuDel}(V)$, contains exactly all locally detectable directed local Delaunay edges in $\mathrm{UDG}(V)$.

Just like PDT, PuDel is symmetric, planar, and connected if applied onto a connected unit disk graph. In addition, a node can compute its adjacency in PuDel by merely checking if its 1-hop neighbors satisfy the PuDel properties. PuDel is a constant stretch Euclidean spanner for the UDG with a Euclidean spanning ratio of at most $((1+\sqrt{5})/4)\pi^2 \approx 7.98$.

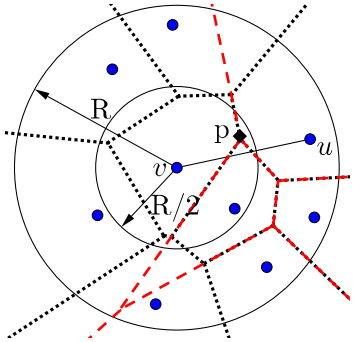


Figure 3.2: Illustration of the two PuDel criteria: $uv \in \mathrm{UDG}(V)$ is contained in $\mathrm{PuDel}(V)$ since the intersection of $\mathrm{VR}_{\mathrm{N}_1(v)}(v)$ (dotted) and $\mathrm{VR}_{\mathrm{N}_1(v)}(u)$ (dashed) is not empty and contains a point p with $\|vp\| \leq \mathrm{R}/2$.

The first proof for this property was presented by Xu et al. in [62]. A revised and extended version of this proof is one of the contributions of this thesis and is given in Section 6.1.2.

Li et al. introduce in [66, 104] a k-locally computable supergraph of the unit Delaunay triangulation (UDel), called *Localized Delaunay triangulation* (LDel), whose definition requires the following auxiliary definition.

Definition 3.6 (k-localized Delaunay triangle)**.** Given three nodes $u, v, w \in \mathrm{UDG}(V)$ that build a clique in $\mathrm{UDG}(V)$, the triangle $\triangle(u, v, w)$ is called a *k-localized Delaunay triangle*, if its circumcircle $\mathrm{C}(u, v, w)$ does not contain any other node $x \in V \setminus \{u, v, w\}$, with $x \in \{\mathrm{N}_k(u) \cup \mathrm{N}_k(v) \cup \mathrm{N}_k(w)\}$, i.e., if the circumcircle (including the boundary) does not contain any k-hop neighbor of u, v, and w.

Definition 3.7 (k-localized Delaunay graph)**.** The k-localized Delaunay graph over $\mathrm{UDG}(V)$, denoted by $\mathrm{LDel}^{(k)}(V)$, contains exactly all edges from $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$, as well as all edges from all k-localized Delaunay triangles.

For any $k \geq 1$, graph $\mathrm{LDel}^{(k)}(V)$ is connected if applied onto a connected UDG. It is planar for $k \geq 2$. Since $\mathrm{UDel}(V) \subseteq \mathrm{LDel}^{(k)}(V)$ for any $k \geq 1$, it has a constant Euclidean spanning ratio of at most 5.08. A node can compute its adjacency in $\mathrm{LDel}^{(k)}(V)$ by merely looking at its $(k+1)$-hop neighborhood.

Since $\mathrm{LDel}^{(1)}(V)$ is not planar, Li et al. [66, 104] propose a local algorithm for construction of the *planar localized Delaunay triangulation* (PLDel), which is a planarized version of $\mathrm{LDel}^{(1)}(V)$. Essentially nodes compute their 1-hop Delaunay triangulations and then exchange triangles of side length at most R with their neighbors. Based on this information, each node can compute its adjacency in $\mathrm{LDel}^{(1)}(V)$ and distribute the adjacency in this graph to their 1-hop neighbors in UDG. Given this information, each node can remove invalid Delaunay triangles. A final distribution of all incident GG-edges and valid Delaunay triangles gives the final result: the adjacency in $\mathrm{PLDel}(V)$. The latter is a planar and connected supergraph of $\mathrm{LDel}^{(2)}(V)$ and therefore, a constant stretch Euclidean spanner for $\mathrm{UDG}(V)$. Auraújo and Rodrigues [63] as well as Bose et al. [114]

show how to compute PLDel more communication efficient, i.e., with fewer communication steps, but their approaches have the same worst-case message complexity of $\mathcal{O}(n)$.

Luan et al. [115] introduce the 1-*hop local Delaunay diagram based on GG with additional crossing edge elimination* (1^+-GLDD). It is similar to $\mathrm{LDel}^{(1)}(V)$. 1^+-GLDD consists of all GG edges and all non-GG edges uv for which it holds that the circumcircle around u, v, and the angle maximizing node w contains no nodes from $\mathrm{N}_1(u) \cup \mathrm{N}_1(v)$. This graph may be non-planar; however, edge intersections can easily be removed by making use of partial two-hop neighborhood information. 1^+-GLDD is a connected and planar subgraph of the underlying UDG and a supergraph of GG. Therefore, its Euclidean spanning ratio is at most as large as that of GG.

Satyanarayana and Rao [116] observe that local planarization techniques like GG and PLDel suffer from large hop-spanning ratios since nodes that are connected in the input UDG may be separated by several hops in the planar subgraph. To remedy this problem, they propose the *Constrained Delaunay Triangulation* (CDT), which is a planar subgraph of the underlying UDG. Essentially they first compute PLDel and add long UDG edges between nodes that are not connected in PLDel. Then, edges from PLDel intersecting with the newly introduced edges are removed, based on local neighborhood information exchange. Their algorithm is 2-local. The authors claim that CDT is a spanner, but do not specify any spanning ratio, in particular not whether it is constant at all.

In a series of publications [89, 117–123], Kanj et al. show various results in the context of local, bounded degree construction of constant stretch Euclidean and constant stretch *Power spanners*.[2] These topologies are either subgraphs of GG (denoted KP06 and KPX09 in Table 3.2) or $\mathrm{LDel}^{(2)}$ (denoted by MYS, KPXLoc08, and Δ_{11}-Spanner in Table 3.2). At this point, only the latest work [123] is further discussed, for it introduces the best results w.r.t. the Euclidean spanning ratio. Therein, Kanj et al. introduce algorithm Δ_{11}-*Spanner*. Given a node from $\mathrm{UDG}(V)$, it locally computes this node's adjacency in a planar subgraph spanner whose Euclidean spanning ratio is < 7 and which has a node degree of 11 at most. The node first determines its adjacency in $\mathrm{LDel}^{(2)}(V)$ and selects any three consecutive and adjacent edges in $\mathrm{LDel}^{(2)}(V)$ whose angles at v sum up to $4\pi/5$. The remaining space is partitioned into a maximal number of cones of apex at most $\pi/5$. In each such cone the node selects the neighbor in $\mathrm{LDel}^{(2)}(V)$ closest to it. For every empty cone, one unselected edge left or right of that cone is selected, based on some additional geometric constraints. Finally, this node keeps a selected edge if the respective endpoint also decides to select it. In order for a node to decide its adjacency in the Δ_{11}-Spanner, it has to know at most all of its 4-hop neighbors. In fact, the 3-hop neighborhood suffices to decide its adjacency in $\mathrm{LDel}^{(2)}(V)$ and to compute the outgoing edges in the Δ_{11}-Spanner. The final verification step for such an edge, however, requires 3-hop neighborhood information of the respective edge's endpoint.

Frey and Rührup [14] propose a concept called *Direct Planarization*. Instead of using geometric properties (like in GG, RNG, or DT) that implicitly construct planar graphs, they describe two schemes which remove edge intersections explicitly: *Angle-based Direct*

[2] For further details on planar Power spanners, their local construction, and other related work on this topic, the reader may refer to [89, 117, 118] and references given therein.

Planarization (ADP) and *Delaunay-based Direct Planarization* (DDP). Given a unit disk graph edge uv, ADP does not remove this edge, if for all edges wx intersecting uv it holds that $\max\{\measuredangle uwv, \measuredangle uxv\} < \max\{\measuredangle wux, \measuredangle wvx\}$. DDP does not remove edge uv if for all intersecting edges wx it holds that x is not contained by the circle $\mathrm{C}(u, v, w)$. The application of these simple geometric rules onto a connected unit disk graph ensures the construction of a connected and planar subgraph. It moreover holds that $\mathrm{GG}(V) \cap \mathrm{UDG}(V) \subseteq \mathrm{ADP}(V)$ and $\mathrm{RDG}(V) \subseteq \mathrm{DDP}(V)$, where RDG denotes the *restricted Delaunay triangulation*, which is introduced shortly. Hence, the Euclidean spanning ratio of ADP is at most as large as that of GG, and DDP is even a constant stretch Euclidean spanner for the underlying unit disk graph, since RDG is a constant stretch Euclidean spanner for it. For a node $v \in V$ to compute its adjacency in $\mathrm{ADP}(V)$ and $\mathrm{DDP}(V)$ it is sufficient to know its 2-hop neighborhood.

The idea of *Direct Planarization* is further elaborated by Mathews and Frey in [124]. They propose the two-staged algorithm *localized link removal and addition based planarization* (LLRAP). In the link removal stage, each node u removes exactly those adjacent links uv for which it holds that they are intersected by another edge wx, with $w \in \mathrm{N}_1(u)$, and simultaneously wv is an edge in the network graph. This stage planarizes the network graph but may lead to disconnection. Therefore, in the second stage, each node u checks for each edge uv which has been removed in the first stage, if uv can be added without producing new intersections. It is added again only if both u and v have ensured that adding edge uv does not lead to an intersection. Nodes can decide their neighborhoods in LLRAP based on their 3-hop neighborhoods. LLRAP is planar and connected if applied onto a connected unit disk graph. Moreover, it is a Euclidean weak spanner with a spanning ratio of at most 3/2. Interestingly, it ensures the aforementioned properties not only for unit disk graphs, but for all input graphs satisfying the *coexistence* and *redundancy* properties, which are defined next.

Definition 3.8 (Coexistence property)**.** A network graph $G = (V, E)$ satisfies the *coexistence property* if for any node triple $u, v, w \in V$ with $\{uv, uw, vw\} \subseteq E$ and any $x \in V$ located inside the triangle $\triangle(u, v, w)$ it holds that $\{xu, xv, xw\} \subseteq E$.

Definition 3.9 (Redundancy property)**.** A network graph $G = (V, E)$ satisfies the *redundancy property* if for any two intersecting edges $uv, xy \in E$ it holds that at least one node $z \in \{u, v, x, y\}$ is connected in G by an edge to the remaining three nodes $\{u, v, x, y\} \setminus \{z\}$.

Unit disk graphs satisfy both the redundancy (see proof of [125, Lemma 4.1]) and the coexistence property (see proof of [124, Lemma 1]).

Localized Approaches Based on Proximity Graphs

The following localized algorithms also produce planar and connected subgraphs of unit disk graphs by means of (Delaunay-based) proximity graphs, but in contrast to the aforementioned ones, they are not local (for definitions of these terms, see Definitions 2.3 & 2.5).

Gao et al. introduce in [125, 126] the *restricted Delaunay graph* (RDG), which is a planar and connected supergraph of the unit Delaunay triangulation (UDel). Therefore,

RDG is a constant stretch Euclidean spanner for the underlying UDG. A node u adds exactly those unit disk graph edges uv to RDG, for which it holds that $uv \in \mathrm{Del}(\mathrm{N}_1(w))$, for all $w \in \mathrm{N}_1(u) \cap \mathrm{N}_1(v)$. Hence, the decision of whether an incident edge of some node belongs to RDG or not depends only on that node's 2-hop neighborhood. Gao et al. do not construct RDG directly on a given unit disk graph, but on a clustered subset of it in order to also obtain a constant topological spanning ratio. For this reason they first apply the localized, hierarchical, round-based clustering scheme from [127, 128], where nodes nominate neighboring nodes with largest IDs as cluster heads until finally a small set of cluster heads is selected. In order to obtain a connected backbone graph, once this clustering has been computed, cluster heads are locally interconnected by gateway nodes. RDG is then constructed on the subgraph induced by the cluster heads and gateway nodes. Finally, non-backbone nodes are connected to their closest cluster heads. The resulting graph has a constant topological stretch factor, as well as a constant Euclidean stretch factor. In the backbone graph, nodes have a constant node degree, whereas after connection of non-backbone nodes to closest cluster heads, the latter may have an unbounded node degree. It is worth noting that the clustering is the only localized and non-local part of the computation. All remaining steps can be performed locally.

Alzoubi et al. [129] propose an approach which combines clustering techniques with the localized Delaunay triangulation (LDel) [66, 104]. They first apply a localized algorithm [130] for construction of a Connected Dominating Set (CDS). This CDS has constant node degree and the graph CDS', resulting from connecting each non-backbone node by an edge with its dominator, is a constant stretch Euclidean and constant stretch topological spanner with spanning ratios of at most 6 and 3, respectively, for the underlying UDG. In particular, the bounded degree property also holds for the graph induced by the node set of CDS w.r.t. the underlying UDG, called ICDS. The latter obeys the UDG properties but is not planar. Therefore, in a second stage, they locally compute PLDel [66, 104] (discussed previously) on top of ICDS. The resulting graph *LDel(ICDS)* is a constant stretch Euclidean as well as a topological spanner for the underlying UDG, and has only a constant node degree. However, as in the case of the approach by Gao et al. [125, 126], the backbone construction itself is localized but not local, whereas all remaining steps can be performed locally.

In [131,132], Wang and Li introduce a localized algorithm for construction of the *bounded degree localized Delaunay triangulation*, called BPS_2. For a given $\mathrm{UDG}(V)$, each node first computes locally its adjacency in $\mathrm{LDel}^{(2)}(V)$ based on complete two-hop neighborhood information and exchange of the two-hop Delaunay triangulation with neighboring nodes. Subsequently in a round-based scheme, some nodes apply a *Yao-step*[3] and select a constant number of their incident LDel edges, whereas other nodes retard their further actions

[3] The Yao_p graph proposed by Yao [133] is a technique for 1-local computation of supergraphs of the MST in high dimensions. Each node partitions its transmission radius in p equally spaced cones centered at it. For each cone, any node u selects the closest neighbor v, if there is any, and adds the directed edge $\overrightarrow{uv}$ to Yao_p. This graph then contains the MST as a subgraph. Hence, deletion of all undirected links yields a connected graph of node degree at most p, but which is not necessarily planar. This technique is often used for local computation of degree bounded subgraphs and is commonly referred to as *Yao-step*.

until certain conditions in the neighborhood apply. Therefore, this algorithm is localized but not local. The resulting graph BPS_2 has a node degree of $19 + \lceil 2\pi/\alpha \rceil$ and Euclidean spanning ratio of $\max\{\pi/2, \pi\sin(\alpha/2) + 1\} \cdot \mathcal{C}_{\mathrm{del}}$, where $0 < \alpha \leq \pi/3$ is a parameter of the algorithm and $\mathcal{C}_{\mathrm{del}}$ denotes the dilation of the Delaunay triangulation.

Based on the same algorithmic idea, Song et al. [134, 135] propose the computation of the planar power spanners *OrdYaoGG* and *SYaoGG*. The former has node a degree of at most $k + 5$, for a parameter $k > 6$ and the latter has a degree of k, for an integer parameter $k \geq 9$. Instead of using $\mathrm{LDel}^{(2)}(V)$, the Yao-step is performed on top of the nodes' Gabriel graph neighborhoods. Due to use of the Gabriel graph, nodes have to exchange messages only with other nodes in their 2-hop neighborhoods. While being subgraphs of GG, these graphs have Euclidean and topological spanning ratios of $\Omega(\sqrt{n})$ and $\Theta(n)$, respectively. Finally, for the same reason as in the approach of Wang and Li [131, 132], these algorithms are also localized but not local (see also the discussion by Kanj et al. [89]).

Clustered Approaches and Planar Overlays

Frey [51] introduces the *purged aggregated Gabriel graph* (PAGG), a connected and planar overlay graph for a given unit disk graph. The plane is partitioned into a regular hexagon grid with grid cell diameter of at most unit disk radius. First, for each node its adjacency in the Gabriel graph is computed. Then, for each non-empty cell one vertex, represented by the cell's geographic center, is introduced. The *aggregated Gabriel graph* connects any two such vertices by an overlay edge, if there exists a Gabriel graph edge which connects two nodes from the respective cells in the underlying network graph. The aggregated Gabriel graph is not planar since it may be the case that the endpoint u of some overlay edge uv may be located on another overlay edge xy. Such *irregular intersections* can easily be handled by removing edge xy and replacing it by two appropriate (virtual) overlay edges, which are edges that are not necessarily present in the aggregated Gabriel graph. The planarized overlay graph is then called a purged aggregated Gabriel graph. Any node can compute the adjacency of its cell's vertex in PAGG using at most 3-hop neighborhood information.

Independently of Frey [51], Tejeda et al. [136,137] describe a similar idea for construction of the so called *virtual spanner* (VS). Given a regular tessellation of the plane (e.g., a hexagonal partitioning) with cell diameter equal to the unit disk radius, initially for every non-empty cell a vertex at this cell's geographic center is introduced. Given a particular vertex v it is first connected by an overlay edge to any neighboring cell which is connected by a unit disk edge. Then, for every cell w which is potentially reachable by a unit disk edge from the cell represented by v, it is checked if adding overlay edge vw causes an intersection; it is added only if this is not the case. This way, a planar and connected overlay graph of the given unit disk graph is constructed. Similarly to the approach from Frey [51], the construction of incident overlay edges of a node's cell requires the node to know at most its 3-hop neighborhood in UDG.

The definition of spanning ratio cannot directly be applied, since overlay graphs are not subgraphs of the underlying network graph. Instead, one can compare the ratio of

the distance traveled by a message when simulating routing on the planar overlay graph to the length of the actual shortest path in the network graph. This ratio represents the *routing stretch* introduced by an overlay graph. However, neither Frey [51] nor Tejeda et al. [136, 137] consider the routing stretch of their approaches.

Catusse et al. [138] also present an approach based on geographical clustering of the plane. In contrast to the aforementioned ones, their algorithm produces an actual subgraph of the network graph. The plane is partitioned by a regular square grid, where the grid cell diameter is equal to the unit disk radius. Cells that are adjacent by an edge in UDG are connected by the shortest edge. Those edges for which there exist short hop-paths are subsequently removed. All remaining intersections are handled by a pattern matching routine: it identifies the type of edge crossing, removes one of the crossing edges, and possibly adds another edge, according to the specific case, in order to avoid disconnection of the graph. In a final step, nodes within the cells are connected as follows. Each cell can be dissected by at most one inter-cluster edge, which partitions the cell into at most two regions and the nodes in each non-empty cell into at most two subsets of nodes. Between any two adjacent regions, the shortest edge connecting them is added. Between any nodes in a subset, which are connected to nodes outside the cell (by inter-cluster edges or inter-partition edges), a minimum spanning tree is constructed. Finally, each node in each subset which is not connected to any node outside its cell is connected to the closest neighbor in the spanning tree of its partition. The resulting graph is a planar, connected subgraph of UDG and has a constant topological spanning ratio of at most $10\alpha + 9$, where α is the maximum distance between any two nodes that belong to the same cell [138]. Although the authors claim that the output graph can be constructed using only local neighborhood information, no algorithmic details are provided. In particular the degree of locality is not clear.

Damian and Pemmaraju [139] introduce the local algorithm PLOS (Planar Localized Spanner) for the construction of planar Euclidean spanners of low weight. In a first step, nodes locally compute their adjacencies in PLDel [66] (see above). The plane is virtually partitioned using a regular square grid of overlapping square cells, such that nodes within one cell form a clique and each node belongs to at most four cliques. This gives a clique cover of the node set. For each such clique, the planar, bounded degree Euclidean spanner BPS_2 [131, 132] (see above) is computed. Since BPS_2 is applied onto a clique, this step is actually local and not localized. Then, each node removes incident edges with both endpoints in the same clique. Finally, a clustering step is used again to filter out those edges which span multiple cliques. The output graph H_{PLOS} is a $\mathcal{C}_{\mathrm{del}} \cdot (1+\epsilon)(1+\pi/2)$-Euclidean spanner for the input UDG, where $\mathcal{C}_{\mathrm{del}}$ refers to the dilation of the Delaunay triangulation. It has a constant node degree and its weight is only a constant times that of the MST.

Other Approaches

So called *local minimum spanning trees* (LMST) have first been introduced by Li et al. [140, 141] and subsequently generalized to *k-local minimum spanning trees* (LMST_k) by Li et al. [142].

LMST_k contains a directed edge $\overrightarrow{uv}$, if uv belongs to the minimum spanning tree over node set $\mathrm{N}_k(u)$, denoted $\mathrm{MST}(\mathrm{N}_k(u))$; i.e., it can be computed by a k-local algorithm. Two undirected variants of this graph can be distinguished. LMST_k^- contains an edge uv if both directed edges $\overrightarrow{uv}$ and $\overrightarrow{vu}$ belong to LMST_k, whereas LMST_k^+ contains edge uv if at least one of these two directed edges is contained by LMST_k. LMST_k and its variants are connected and planar subgraphs of the underlying unit disk graph and have a maximum node degree of 6. Moreover, both LMST_k^- and LMST_k^+ are subgraphs of RNG and EMST is a subgraph of LMST_k for any $k \geq 1$. The former implies that LMST_k^- and LMST_k^+ have Euclidean spanning ratios which are at least as large as that of RNG. In addition, a topological spanning ratio of $\Theta(n)$ follows.

Li et al. [142] then further combine the ideas of local minimum spanning trees and the degree bounded graph RNG' [112] (see above) and obtain the *Incident MST and RNG Graph* (IMRG). Its definition requires the following notation.

$$\mathrm{N}_2^{\mathrm{RNG}'}(u) = \{w : vw \in \mathrm{RNG}' \text{ and } v \in \mathrm{N}_1(u)\} \cup \mathrm{N}_1(u)\,.$$

IMRG contains a directed edge $\overrightarrow{uv}$ if edge uv belongs to $\mathrm{EMST}(\mathrm{N}_2^{\mathrm{RNG}'}(u))$. The two undirected variants IMRG^- and IMRG^+ are then defined symmetrically to the undirected variants of LMST_k. IMRG is a subgraph of RNG'. Thus, IMRG and its undirected variants are planar and have a bounded node degree. In addition, IMRG^+ and IMRG^- contain EMST as a subgraph and are therefore connected. IMRG and its undirected counterparts can be computed by a 2-local algorithm, and their Euclidean and topological spanning ratios are at least as large as those of RNG'.

Wattenhofer and Zollinger [143] introduce the 2-local algorithm XTC, which is aimed at operation on network graphs that do not necessarily obey unit disk graph properties. Initially all nodes gather their 1-hop neighborhood information and compute a local order of their incident links. A link to a neighbor v appearing early in the order of u is regarded as being of higher quality than the link to a neighbor w placed later in that order [143]. Once the order is computed, each node exchanges the order with all 1-hop neighbors. Then, each node u traverses its order with decreasing link quality and selects the currently processed neighbor v, only if it has no neighbor w, which appears earlier in u's order and which appears earlier in v's order than u. If executed on a unit disk graph, then the output graph G_{XTC} is symmetric, connected, planar, and a subgraph of the respective RNG. In case the unit disk graph contains no node which has two or more neighbors at exactly the same distance, then G_{XTC} is even identical to the respective RNG. From this it follows that G_{XTC} has Euclidean and topological spanning ratios which are at least as large as those of RNG. On the other hand, this graph has a node degree of at most 6. Algorithm XTC works correctly in general weighted graphs without geometric assumptions. Under this general model, however, the graph is only guaranteed to be symmetric.

Li et al. [67, 144] propose a radically new family of planar geometric graphs called *Hypocomb*. A Hypocomb graph is the dual of a truncated mesh, also referred to as *Blocked-mesh* or simply *Besh*. For a given set of nodes, the Besh is constructed as follows. For each node draw four rays in each cardinal direction and allow distance-based blocking

if two rays intersect. Hypocomb is obtained by connecting any two nodes that have a ray-blocking relation in the Besh. It is a connected planar graph with an unbounded node degree for the complete Euclidean graph. If instead only those edges are added between nodes that have a mutual ray-blocking relation, *Reduced Hypocomb* is obtained, which remains connected and planar, but has a node degree of six at most. Most importantly, a variant of Reduced Hypocomb, called *Local Hypocomb* (LHC), can be computed by a 1-local algorithm. For any incident edge uv, node u simply computes if uv is an edge in the Reduced Hypocomb w.r.t. node set $\mathrm{N}_1(u) \cap \mathrm{N}_1(v)$. The set of all such edges constitutes Local Hypocomb, which is a symmetric, planar, connected subgraph of node degree of at most 8 of the underlying UDG. However, it is unknown if graphs of the Hypocomb graph family satisfy constant spanning properties.

Taxonomy of UDG Planarization Techniques

Tables 3.1 & 3.2 summarize the above given survey on unit disk graph planarization techniques. Table 3.1 shows the resulting subgraph relations. Each of the rows 2–16 relate only to row 1; e.g., row 3 shows that EMST $\subseteq$ LMST $\subseteq$ RNG, which then implies further subgraph relations of RNG as given by row 1. Those topologies for which it is only known that they are subgraphs of the unit disk graphs are omitted. Table 3.2 gives a complete taxonomy in chronological order. For each topology control structure the following properties are listed:

k-locality refers to the degree of k-locality required by nodes to compute their adjacency.
Local is marked '✓' if the structure is computed by a k-local algorithm and '✗' otherwise.
Graph type specifies whether this structure is a subgraph (S) over all network nodes or an overlay graph (O).
Euclidean stretch specifies the Euclidean spanning ratio if known and '?' otherwise. Constants are provided whenever possible. In all remaining cases, the best known upper and/or lower bounds are given, some of which follow from the subgraph relations given in Table 3.1.
Topological stretch specifies the topological spanning ratio. In case no non-trivial upper bound is known, it is stated to be $\mathcal{O}(n)$, which is the trivial upper bound for any connected graph. Some of the lower bounds are implications of the subgraph relations given in Table 3.1
Node degree is the maximum node degree in the structure, where "$\geq x$" means that the node degree is a constant greater than or equal to x.
References refers to source(s).
Year lists the earliest point in time of publication.

1.	EMST	$\subseteq$	IMRG	$\subseteq$	RNG’	$\subseteq$	RNG	$\subseteq$	CNG	$\subseteq$	GG	$\subseteq$	PDT	$=$	PuDel	$\subseteq$	PDT2	$\subseteq$	UDel	$\subseteq$	$\text{LDel}^{(k)}$	$\subseteq$	...	$\subseteq$	$\text{LDel}^{(2)}$	$\subseteq$	PLDel	$\subseteq$	$\text{LDel}^{(1)}$	$\subseteq$	UDG
2.		$\subseteq$	LRNG	$\subseteq$																											
3.		$\subseteq$	LMST			$\subseteq$																									
4.					G_{XTC}	$\subseteq$																									
5.									OrdYaoGG	$\subseteq$																					
6.									SYaoGG	$\subseteq$																					
7.									KP06	$\subseteq$																					
8.									KPX09	$\subseteq$																					
9.												$\subseteq$	MG																	$\subseteq$	
10.												$\subseteq$	1^{+}-GLDD																	$\subseteq$	
11.												$\subseteq$	ADP																	$\subseteq$	
12.																			$\subseteq$		RDG	$\subseteq$	DDP							$\subseteq$	
13.																							BPS_2	$\subseteq$							
14.																							MYS	$\subseteq$							
15.																							KPXLoc08	$\subseteq$							
16.																							Δ_{11}-Sp.	$\subseteq$							

Table 3.1: Summary of unit disk subgraph relations.

Topology	k-locality	Local	Graph type	Eucl. stretch	Top. stretch	Node degree	Ref.	Year
GG	1	✓	S	$\Theta(\sqrt{n})$	$\Theta(n)$	$\mathcal{O}(n)$	[105]	1969
RNG	1	✓	S	$\Theta(n)$	$\Theta(n)$	$\mathcal{O}(n)$	[106]	1980
RDG	2	✗	S	≤ 5.08	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[125, 126]	2001
LDel$^{(k)}$	$k+1$	✓	S	≤ 5.08	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[66, 104]	2002
PLDel	2	✓	S	≤ 5.08	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[66, 104]	2002
PDT	1	✓	S	≤ 7.98	$\Theta(n)$	$\mathcal{O}(n)$	[49, 50]	2002
PDT$_2$	2	✓	S	≤ 7.98	$\Theta(n)$	$\mathcal{O}(n)$	[49, 50]	2002
LMST$_k^{(+/-)}$	$k \geq 1$	✓	S	$\Omega(n)^\dagger$	$\Theta(n)^\dagger$	≤ 6	[140–142]	2003
LDel(ICDS)	3	✗	S	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	[129]	2003
RNG'	1	✓	S	$\Omega(n)^\dagger$	$\Theta(n)^\dagger$	≤ 6	[112]	2003
LRNG	2	✓	S	$\Omega(n)^\dagger$	$\Theta(n)^\dagger$	≤ 6	[112]	2003
BPS$_2$	3	✗	S	< 6.22	$\mathcal{O}(n)$	≥ 25	[131, 132]	2003
G_{XTC}	2	✓	S	$\Omega(n)^\dagger$	$\Theta(n)^\dagger$	≤ 6	[143]	2004
OrdYaoGG	2	✗	S	$\Omega(\sqrt{n})^\dagger$	$\Theta(n)^\dagger$	≥ 12	[134, 135]	2004
SYaoGG	2	✗	S	$\Omega(\sqrt{n})^\dagger$	$\Theta(n)^\dagger$	≥ 9	[134, 135]	2004
MG	2	✓	S	$\mathcal{O}(\sqrt{n})^\dagger$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[111]	2004
1^+-GLDD	2	✓	S	$\mathcal{O}(\sqrt{n})^\dagger$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[115]	2004
IMRG$^{(+/-)}$	2	✓	S	$\Omega(n)^\dagger$	$\Theta(n)^\dagger$	≤ 6	[142]	2004
PAGG	3	✓	O	-	-	$\mathcal{O}(1)$	[51]	2005
KP06	2	✓	S	$\Omega(\sqrt{n})^\dagger$	$\Theta(n)^\dagger$	≥ 13	[117]	2006
VS	3	✓	O	-	-	$\mathcal{O}(1)$	[136, 137]	2006
MYS	4	✓	S	3.54	$\mathcal{O}(n)$	≥ 14	[118–120]	2007
CDT	2	✓	S	?	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[116]	2008
CNG	1	✓	S	$\Theta(n)$	$\Theta(n)$	$\mathcal{O}(n)$	[18, 107]	2008
KPXLoc08	≤ 26	✓	S	8.81	$\mathcal{O}(n)$	≤ 14	[120, 121]	2008
ADP	2	✓	S	$\mathcal{O}(\sqrt{n})^\dagger$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[14]	2009
DDP	2	✓	S	≤ 5.08	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[14]	2009
KPX09	2	✓	S	$\Omega(\sqrt{n})^\dagger$	$\Theta(n)^\dagger$	≥ 10	[89]	2009
Catusse	?	✓	S	?	$10\alpha+9$	$\mathcal{O}(n)$	[138]	2010
H_{PLOS}	≥ 2	✓	S	$\mathcal{O}(1+\epsilon)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	[139]	2010
LHC	1	✓	S	?	$\mathcal{O}(n)$	≤ 8	[67, 144]	2011
PuDel	1	✓	S	≤ 7.98	$\Theta(n)$	$\mathcal{O}(n)$	[62]	2011
LLRAP	3	✓	S	?	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[124]	2012
Δ_{11}-Sp.	4	✓	S	< 7	$\mathcal{O}(n)$	≤ 11	[123]	2012

†: Implication of subgraph relations which are listed in Table 3.1.
Graph type: S=Subgraph and O=Overlay graph for the underlying network graph.
RDG refers to the variant without clustering.

Table 3.2: Taxonomy of unit disk graph planarization techniques in chronological order.

3.2 Algorithms for Quasi Unit Disk Graphs

Next, a comprehensive review of related work on local and localized planarization and spanner construction techniques under QUDG model assumptions is given. This includes some important aspects regarding geographic routing. Some of these algorithms require that the ratio of the maximum to the minimum transmission radius in the underlying QUDG is bounded by $\sqrt{2}$, i.e., $\mathrm{R/r} \leq \sqrt{2}$. In this case a QUDG is said to be *restricted*. Otherwise, it is said to be *arbitrary*.

Quasi Unit Disk Graph Properties

Fact 3.10. Given a connected quasi unit disk graph $G = (V, E)$ with transmission radii r and R such that $\mathrm{r} < \mathrm{R}$, in general it is impossible to planarize G by simply removing edges while preserving graph connectivity.

To see this, consider Figure 3.3a showing a connected QUDG with $\mathrm{r} < \mathrm{R}$, where neither of the two intersecting edges can be removed without disconnecting the graph.

This observation implies in particular that the local, proximity-based planarization techniques introduced in the context of UDG, such as the Gabriel graph or the partial Delaunay triangulation, applied onto arbitrary QUDGs with $\mathrm{r} < \mathrm{R}$, possibly yield disconnected output graphs.

Fact 3.11. If the ratio $\mathrm{R/r} > \sqrt{2}$, then for any integer k there exists a QUDG G such that the existence of an edge intersection in G cannot be detected by any k-local algorithm.

This was first observed by Barrière et al. [83, 145]. See Figure 3.3b for an illustration. Under the assumption that the ratio $\mathrm{R/r} > \sqrt{2}$, it is possible to construct example graphs, where any path between the endpoints of an edge uv and those of an intersecting edge xy consists of more than k hops; hence, any k-local algorithm fails to detect this intersection.

In contrast, in case of restricted QUDGs, i.e., graphs for which $\mathrm{R/r} \leq \sqrt{2}$ holds, edge intersections can always be detected locally by the nodes incident to it.

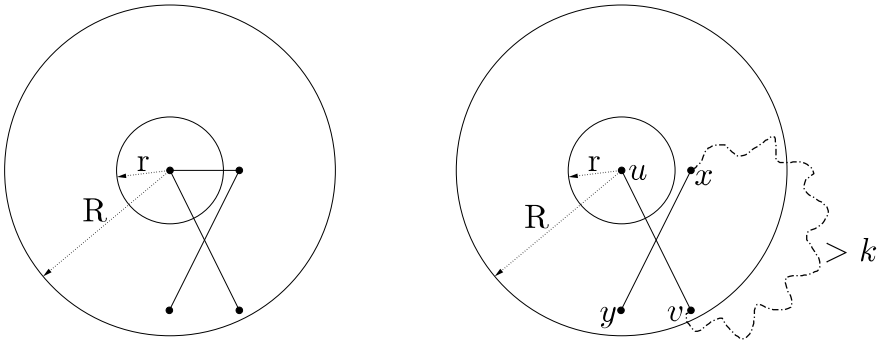
Lemma 3.12. *Let $G = (V, E)$ be a QUDG with $\mathrm{R/r} \leq \sqrt{2}$. If two edges $uv, xy \in E$ intersect, then there exists $wz \in E$ with $w \in \{u, v\}$ and $z \in \{x, y\}$.*

This lemma was previously proven by Kuhn et al. in [86, Lemma 8.2] and Lillis et al. in [71, Lemma 1] and it implies the following fact.

Fact 3.13. The maximum hop distance between the endpoints of two intersecting edges uv, xy in a QUDG G satisfying $\mathrm{R/r} \leq \sqrt{2}$ is at most three. Hence, 3-hop neighborhood information in G is sufficient for a node to determine all edges and their corresponding endpoints, which intersect its adjacent edges in G.

Guan [73, Lemma 6.2] further extends this fact by proving the following lemma.

Lemma 3.14. *Let $G = (V, E)$ be a QUDG with $\mathrm{R/r} \leq \sqrt{2}$ and $uv, xy \in E$ two intersecting edges. If xy intersects uv at a point between u and the midpoint of uv, then there exists $uz \in E$ with $z \in \{x, y\}$.*



(a) (b)

Figure 3.3: (a) Example of a QUDG that cannot be planarized without disconnecting it. (b) Example of a QUDGs where the edge intersection between uv and xy cannot be detected by any k-local algorithm since the endpoints in $\{u, v\}$ are more than k hops apart from the endpoints in $\{x, y\}$.

This lemma implies the following fact.

Fact 3.15. A node in a QUDG G satisfying $\mathrm{R/r} \leq \sqrt{2}$, which is provided with 2-hop neighborhood information in G, knows all edges and their corresponding endpoints, which intersect its adjacent edges uv in G in points that are at least as far away from v than from u.

The aforementioned facts as well as their implications have led to numerous distributed approaches for construction of planar and almost planar topologies and geographic routing protocols that guarantee message delivery if applied thereon. These approaches are subsequently summarized.

Overlay Edge-Assisted Gabriel Graph Planarization

Barrière et al. [83, 145] propose a three-phased algorithm for geographic routing with guaranteed delivery for restricted QUDGs. Starting from the observation that Gabriel graph planarization possibly yields disconnected and asymmetric output graphs if applied onto QUDGs (see example given in Figure 3.4), the *Completion Phase* first adds virtual links between certain nodes in order to obtain a supergraph $S(G)$. These virtual links have a length of at most R and ensure that the output graph $\mathrm{GG}(S(G))$, which is computed during the *Extraction Phase*, does not lose connectivity and stays symmetric. In the *Routing Phase*, local message forwarding using a guaranteed delivery routing algorithm (e.g., GFG [36]) is performed on $\mathrm{GG}(S(G))$. Forwarding along a virtual link is performed by routing the message along an appropriate sequence of incident physical edges. Despite its simplicity and the fact that this protocol guarantees correctness even in fully asynchronous systems, it has several drawbacks. Unless the input graph is *civilized*[4],

[4] A *civilized graph* (the λ-*precision*/$\Omega(1)$ *model*) is defined to be a graph $G = (V, E)$ embedded in $\mathbb{R}^2$, where for any fixed $\lambda > 0$, two nodes $u, v \in V$ are of a distance at least λ apart [146].

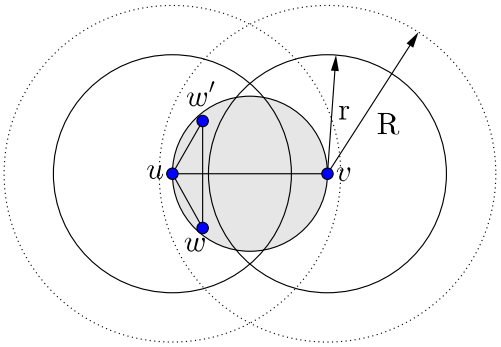


Figure 3.4: Example where the Gabriel graph applied onto a QUDG with minimum and maximum communication radii r and R is disconnected/asymmetric. Node u removes edge uv due to GG-rule violation by its neighbors w and w', whereas node v does not remove this edge since $\mathrm{D}(v,u)$ (shaded area) is empty of neighbors from $\mathrm{N}_1(v) \setminus \{u,v\}$. This figure is similar to [145, Figure 7].

the construction of virtual edges is not necessarily a local operation. This is due to the fact that the addition of a virtual edge may cause the addition of another virtual edge, and so on. Given a λ-civilized graph, then the construction is $(1 + (\mathrm{R}^2 - \mathrm{r}^2)/\lambda^2)$-local. Barrière et al. [83, 145] show that the topological spanning ratio of the output graph is $\Omega(n)$. Currently, there are no results regrading this construction's Euclidean spanning ratio. It is likely that the Euclidean spanning ratio is not constant, because GG is used for planarization. An extension of this algorithm that yields output graphs with shorter Euclidean spanning ratio is one of the contributions of this thesis and is presented in Chapter 9.

In order to reduce the message complexity and avoid construction of unnecessary virtual edges in the aforementioned approach, Moaveninejad et al. [147, 148] suggest to remove short non-Gabriel edges before constructing virtual edges. Only when intersections are actively determined, virtual edges are added. This scheme also yields connected and planar graphs, but it is unknown if these output graphs are equivalent to those obtained by using the approach from Barrière et al. [83, 145]. By means of simulations Moaveninejad et al. compare their approach to the one by Barrière et al. They show that less message transmissions are required for low and medium network densities, whereas more message transmissions are required for high network densities. Moreover, these simulations suggest that the topological as well as the Euclidean spanning ratios of these output graphs are identical, at least on average.

Backbones, Planar Backbones, and Virtual Planar Backbones

Kuhn, Wattenhofer, and Zollinger [84, 86] present localized algorithms for construction of (planar) routing backbones for both arbitrary as well as restricted QUDGs.

For arbitrary QUDGs, at first they let the nodes compute a *maximal independent set* (MIS) and connect the *independent set* (IS) nodes via bridge nodes.[5] This results in the *dense backbone graph*. For further sparsification, they obtain a virtual overlay as follows. Any two of the IS nodes are connected via a virtual edge, if they are connected via a bridge node in the dense backbone graph. The plane is divided by four regularly shifted grids into overlapping square cells of bounded side length. Then, in each cell a standard (centralized) spanner algorithm is applied, which removes some of the virtual edges. The routing backbone G_{BG} is then the sparsified graph with virtual edges replaced by corresponding physical edges. This graph is a $\mathcal{O}(\log(1/(\mathrm{r}/\mathrm{R})))$ hop spanner for the dense backbone graph, but it is not planar. This algorithm is localized but not local since it makes use of the localized MIS computation technique from [130].

For restricted QUDGs Kuhn et al. [84, 86] describe a related protocol. Initially, again an MIS is computed by the nodes and connected via bridge nodes. Then each node determines its incident edges in the induced subgraph on this node set. This gives a sparse QUDG on which the three-phased scheme by Barrière et al. [83,145] can be applied. Due to the clustering, virtual edge construction requires local communication along paths consisting of at most $\mathcal{O}(\Delta^2)$ hops, where Δ is the maximum node degree of the sparse QUDG. The resulting routing graph $GG(S(G'_{BG}))$ is a constant stretch hop spanner for the input graph having constant node degree. According to [71], these approaches do not provide constant Euclidean spanning properties. As an alternative to the application of the protocol from Barrière et al., Kuhn et al. further propose to replace edge intersections of the induced subgraph by virtual nodes in order to planarize it. This idea is later extended by Lillis et al. [70, 71] as described next.

Lillis, Pemmaraju, and Pirwani [70, 71] further elaborate on the virtual nodes idea by Kuhn et al. [84, 86]. Their algorithm works on restricted QUDGs G. First, a backbone graph is constructed in a network-wide initialization phase. Essentially, nodes locally compute a Connected Dominating Set (CDS) on the basis of a regular square-grid geographic clustering, where grid length is chosen dependent on the minimum transmission range, such that all nodes in a cell form a clique. The subgraph G_b induced by the nodes in the CDS w.r.t. G obeys the QUDG properties, has a constant node degree, and has a constant number of edge intersections per edge. The planar overlay graph $Virt(G_b)$ is obtained by replacing every edge intersection in G_b by a virtual node which is controlled by a physical node being incident to it. Nodes in $Virt(G_b)$ have constant node degree. The set of edges connecting non-backbone nodes to their corresponding cluster heads and the planarized backbone graph $Virt(G_b)$ form the routing graph G_r. It is a $(4c+6)$-hop spanner for the input graph, where $c \in \mathcal{O}(1)$ denotes the maximum number of intersections of any edge in G_b. Since this routing graph is not a constant stretch Euclidean spanner, Lillis et al. propose the following extension of the routing graph construction. Instead of using one grid, three shifted grids are placed on the input graph and for each cell the localized Euclidean spanner construction from Wang and Li [132] (see Section 3.1) is employed. The routing graph G'_r then consists additionally of all the grid cell's Euclidean

[5] Given a graph $G = (V, E)$, an *independent set* is a set of nodes $U \subseteq V$, no two of which are adjacent in G. An independent set is *maximal* if no node can be added to it without violating its independence [87].

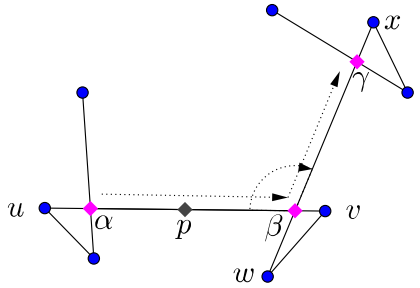


Figure 3.5: Illustration for explanation of protocol Virtual-Face-Traversal by Guan [72,73]. Blue and pink points represent real and virtual nodes, respectively. Point p is the midpoint of edge uv. The dotted arrows show the virtual routing path $\langle \alpha \to \beta \to \gamma \rangle$. This figure is similar to [72, Figure 2]

spanner edges. Depending on the destination, routing is performed either on the planar Euclidean spanners or on the planar backbone graph. Routing graph G'_r is a constant stretch Euclidean spanner, but it is not planar. One major contribution of this thesis is a reactive version of this algorithm. It is presented in Chapter 8.

Guan [72,73] elaborates the virtual node idea even further and presents a local geographic routing protocol, called *Virtual-Face-Traversal*, which enables guaranteed delivery routing in restricted QUDGs without network-wide planarization. Essentially, this protocol simulates FACE routing on the input graph G where edge intersections are considered as virtual nodes.[6] Face routing applied onto the virtual planar graph yields a *virtual routing path* consisting of virtual edges where either none, one, or both of the endpoints are virtual nodes. *Virtual-Face-Traversal* computes a *real path* in the network graph that follows this virtual routing path. This is best explained by means of an example, provided in Figure 3.5. Suppose the endpoint u of real edge uv, which contains the current virtual edge $\alpha\beta$, holds the packet containing α as the endpoint of the current virtual edge. Given that u is provided with 2-hop neighborhood information in G and that the endpoint β is in between u and the midpoint p of uv, then u can determine β and compute the real path $u \to v \to w$ (according to Fact 3.15). It then forwards the packet to w containing β as the endpoint of the current virtual edge. Otherwise, if β is in between p and v, then u forwards the packet to v containing α as the endpoint of the current virtual edge. Based on its 2-hop information, v is now able to determine the endpoint β of the current virtual edge and forwards the packet to w containing β as the endpoint of the current virtual edge. In either case, node w takes care of computation of the next virtual routing step and simulation of the underlying real path.

Virtual-Face-Traversal is a 2-local protocol for guaranteed delivery geographic routing, which does not require topology control operations. However, by avoiding topology control and performing routing on the actual network graph, where the number of edge

[6] In contrast to the original FACE routing algorithm, where a face is changed whenever the start-to-destination-line (stored in the packet header) is about to be intersected, Virtual-Face-Traversal makes a face change and updates the starting point, whenever a point (not necessarily a node) on a forwarding edge is detected which is closer to the destination than the current starting point.

intersections per edge is not bounded by a constant, this approach can cause severe overhead regarding the number of message transmissions. One solution to this problem is the combination of this approach with the approach by Lillis et al. [70, 71] described above. This is part of the contributions of this thesis and is presented in Chapter 8.

Chen et al. show in [149, 150] how to locally compute *well separable*, nearly-planar Power stretch spanners for arbitrary QUDGs. As in the case of the two approaches by Kuhn et al. [84, 86] and Lillis et al. [70, 71], they make use of a backbone, based on the geographic grid clustering techniques. A *(vertex) separator* of a graph is a subset of vertices that splits the graph into two non-adjacent components of similar size when it is removed from the input graph [150]. A graph is called *well separable*, when any subgraph has a relatively small separator [150]. The backbone is constructed as follows. For the subgraph consisting only of edges of length at most r, a non-connected planar, bounded-degree, Power stretch spanner is computed locally following the idea of [117] (involving Gabriel graph planarization and a Yao-step for bounding the node degree). Then, a grid of constant cell size is imposed and the shortest edges connecting two cells are added to the backbone graph. The output graph $\mathrm{QUDG}_{\mathrm{BB}}$ is then a well separable, nearly-planar (the average number of edge-crossings per edge is bounded by a constant), $\mathcal{O}(\mathrm{R}^2/\mathrm{r}^2)$ degree-bounded, $3 + 2^{\beta+1}\sin^{\beta}(\pi/k)$-power spanner for the input QUDG, where $k \geq 9$ is a parameter and β is the power exponent.

Funke and Milosavljević [151] propose a landmark-based scheme for geographic routing with guaranteed delivery on restricted QUDGs. It operates on a "macroscopic" level of abstraction. The main idea is based on ideas from GLIDER [152] and can be summarized as follows. At first, a k-hop independent set of landmark nodes is computed in a localized fashion. From this the *graph Voronoi diagram* can be obtained. Each landmark node represents a Voronoi cell. Each Voronoi cell consists of all regular nodes (i.e., non-landmark nodes) for which this landmark is closest w.r.t. hop-distance. The *combinatorial Delaunay graph* (CDG) is then the graph over all landmark nodes, where any two nodes are connected by an edge if their corresponding Voronoi cells are connected by an edge in the underlying network graph. CDG is typically not planar and is therefore transformed into the *planar Combinatorial Delaunay Map* (CDM) as follows. Any two landmarks are connected by an edge, if there exists a path between them, such that the path itself as well as the 1-neighborhoods of nodes along this path belong to the corresponding two Voronoi cells (see [151, 153] for details). Finally, an embedding of CDM is computed which requires $2k$-local communication. The graph embedding obtained then permits use of guaranteed delivery geographic routing algorithms, such as GFG [36]. However, according to an example construction by Lillis et al. (see [71, Figure 3]), CDM may not be connected. As yet, spanning ratios of CDM have not been considered.

Subgraph Spanners

Chávez et al. [154] introduce a k-local algorithm for construction of connected, spanning subgraphs of arbitrary QUDGs. In fact, they show that the k-local minimum spanning tree construction by Li et al. [142] (see Section 3.1) also applies to QUDGs. Each node gathers k-hop neighborhood information and computes the *Minimum Spanning Tree*

(MST) over this neighborhood set. An edge uv is added to the output graph $G_k^{\prec}$ if both u and v use it in their locally computed MSTs ($\prec$ refers to a unique linear order defined over the edges in order to obtain uniquely defined MSTs). Graph $G_k^{\prec}$ is a connected subgraph of the input graph. It is planar if the input graph is civilized for $\lambda \geq \sqrt{1-\mathrm{r}^2}$. In case a UDG is given, the node degree is bounded from above by 5, whereas in case of arbitrary QUDGs it is bounded by $3 + \frac{6}{\pi \mathrm{r}} + \frac{\mathrm{r}+1}{\mathrm{r}^2}$. Moreover, if $G_k^{\prec}$ is planar and $k\mathrm{r} > 1$, then the weight of this graph is only a constant times higher than that of the MST. Constant upper bounds on the Euclidean or topological spanning ratios of this construction are not known.

Damian et al. [155] propose a centralized algorithm SLISE for construction of low-interference spanners on arbitrary QUDGs and show that it can be turned into a localized algorithm. In the following, only the centralized algorithm is briefly outlined. In a first step, edges are sorted according to their outgoing interference (the outgoing interference of an edge uv is modeled as the number of nodes w with $\|wu\| \leq \|uv\|$ or $\|wv\| \leq \|uv\|$). Then the algorithm greedily eliminates edges with highest interference from the set of candidate spanner edges, until no more edges can be eliminated without violating the spanner property [155]. Afterwards, algorithm *RelaxedGreedy* is applied onto the resulting subgraph, which processes the edges of the input graph in several phases: First, edges that can be replaced by short paths are removed. Then, a distinct clustering of the nodes is computed and only short edges between clusters are chosen. Finally, redundant edges which are geographically close by are removed. For any $t > 1$ and $\epsilon > 0$, the output graph H_{SLISE} is a (possibly non-planar) Euclidean $t(1+\epsilon)$ spanner with constant node degree. Its distributed construction requires $\mathcal{O}(\log(\Delta) + \log^*(n))$ synchronous communication rounds, where Δ denotes the ratio of the maximum to the minimum edge length in the input graph.

In [139], Damian and Pemmaraju improve the former result by introducing a local algorithm for construction of (possibly non-planar) low-weight Euclidean spanners for arbitrary QUDGs. Again, they describe a centralized algorithm called LOS (localized spanner) and show that it can be computed efficiently in a local fashion. As in the case of PLOS (see Section 3.1), at first a partially overlapping square grid is virtually imposed on the plane, such that each cell forms a clique and each node belongs only to at most four cliques. Then, each node determines to which grid cells it belongs as well as to which grid cells its one-hop neighbors belong. For each of the cliques a node belongs to, it computes a Euclidean $(1+\epsilon)$-spanner using a centralized algorithm. Afterwards, each node performs a Yao-step and selects for each of the $k > 8$ cones the shortest edge (if any). This is succeeded by a reverse Yao-step, where any node u removes in each cone any incident Yao-edge created by another node v, except the shortest one. Finally, a filtering is applied on the edges in order to remove all but a constant number of edges incident to a grid cell. This is done via local computation of an r-cluster cover. Basically, in a four-phased algorithm in each cell a cluster center is elected. Nodes which are connected to a cluster center by a path of Euclidean length of at most r belong to this cluster center. Uncovered nodes with largest IDs decide to become a cluster center as well. Then, for any two clusters of the r-cluster cover, one arbitrary edge connecting these clusters is

added to the spanner. The output graph H_{PLOS} is then a Euclidean $(1+\epsilon)$-spanner of the input graph whose weight is only a constant times that of the MST. This output graph has a constant node degree.

Graph Embeddings

Sarkar et al. [156] present a surprisingly different approach to the problem of guaranteed delivery geographic routing on restricted QUDGs. Instead of computing a planar graph that supports recovery for local geographic routing algorithms, a conformal map of the underlying network is computed in a localized fashion. Based on this, a virtual coordinate system is set up such that Greedy routing always succeeds. At first, they extract a planar triangulation of the network graph, namely by locally computing the restricted Delaunay graph (RDG) (see Section 3.1) on the edges of the input graph of length at most r. The resulting graph can be disconnected but can be reconnected by adding virtual or actual QUDG edges of length $>$ r to RDG. The resulting graph is connected and there exists a planar embedding of this graph in the classical, graph-theoretic sense (see Definition 2.9). Then all internal triangles and all boundary regions of the network are oriented, which is necessary to obtain a manifold on which the conformal mapping can be applied. Now each node locally computes a *discrete Ricci flow* (a concept from differential geometry) as well as its virtual coordinates by flattening of the triangles using a distributed gossip-style process that makes use of the output of Ricci Flow computations. Finally, a conformal mapping can be computed. Simulations suggest that the routing stretch is at most 3.21.

Taxonomy of QUDG Planarization Techniques

Table 3.3 summarizes the above given survey on quasi unit disk graph planarization techniques in chronological order. For each topology control structure the following properties are listed:

QUDG indicates if the ratio R/r is arbitrary (A) or restricted (R), i.e., R/r $\leq \sqrt{2}$.

k-loc. refers to the degree of k-locality required by nodes to compute their adjacency. *unb.* abbreviates the word *unbounded*. "$\geq x$" refers to a constant which is greater than or equal to x.

Local is marked '✓' if the structure is computed by a k-local algorithm and '✗' otherwise.

Graph type specifies if this structure is a subgraph (S) over all nodes, overlay graph (O), and/or backbone graph (B) over a subset of the nodes.

Euclidean stretch specifies the Euclidean spanning ratio if known and '?' otherwise.

Topological stretch specifies the topological spanning ratio if known and '?' otherwise. In case of subgraphs of the underlying QUDG, if no upper bound is known, then the trivial upper bound of $\mathcal{O}(n)$ is specified, which holds for any connected subgraph.

Node degree is the maximum node degree in the structure.

References refers to source(s).

Year lists the earliest point in time of publication.

Topology	QUDG	k-loc.	Local	Graph type	Eucl. stretch	Top. stretch	Node degree	Ref.	Year
GG($S(G)$)	R	unb.	✗ [a]	O	?	$\Omega(n)$	$\mathcal{O}(n)$	[83, 145]	2001
G_{BG}	A	≥ 3	✗	S,B	?	$\mathcal{O}(1)$	$\mathcal{O}(1)$	[84, 86]	2003
$GG(S(G'_{BG}))$	R	$\mathcal{O}(\Delta^2)$	✗	O,B	?	$\mathcal{O}(1)$	$\mathcal{O}(1)$	[84, 86]	2003
GG($S(G)$)	R	unb.	✗ [a]	O	?	?	$\mathcal{O}(n)$	[147, 148]	2004
$G_k^{\prec}$	A	≥ 2	✓	S	?	$\mathcal{O}(n)$	$\mathcal{O}(1)$	[154]	2006
QUDG_{BB}	A	≥ 9	✓	S,B	?	$\mathcal{O}(n)$	$\mathcal{O}(1)$	[149, 150]	2007
CDM	R	≥ 2	✗	O,B	?	?	$\mathcal{O}(n)$	[151]	2007
$Virt(G_b)$	R	3	✓	O,B	?	$\mathcal{O}(1)$	$\mathcal{O}(1)$	[70, 71]	2007
G'_r	R	3	✓	O	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	[70, 71]	2007
H_{SLISE}	A	$\mathcal{O}(\log^* n)$	✗	S	$t(1+\epsilon)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	[155]	2009
H_{PLOS}	A	≥ 2	✓	S	$1+\epsilon$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	[139]	2010

Graph type: S=Subgraph, B= Backbone, O=Overlay graph for the underlying network graph
[a] Algorithm is $1 + (\text{R}^2 - \text{r}^2)/\lambda^2$-local for the class of λ-civilized graphs.

Table 3.3: Taxonomy of QUDG planarization techniques in chronological order.

Chapter 4

Survey on Beaconless Algorithms

This survey systematically and comprehensively reviews beaconless algorithms. It is not limited to beaconless algorithms that directly relate to the algorithmic contributions given in this thesis, but shall provide a more general overview on this research field. The scope of this survey is, however, limited to local(ized) beaconless algorithms that do not make use of flooding. Flooding-based algorithms annihilate the advantages of the beaconless approach and are therefore of no further relevance for this thesis.

Section 4.1 introduces the general idea of beaconless algorithms. In Sections 4.2–4.5 the existing algorithms are presented according to their specific objectives. Each section is complemented by a taxonomy that lists and summarizes the work presented therein.

4.1 Preliminaries

Beaconless algorithms are distributed algorithms that distinguish themselves from conventional distributed algorithms by avoiding maintenance of neighborhood information via *beaconing*. The latter refers to the periodic process of sending "*HELLO*"-messages (so called *beacons*) by the network nodes in order to establish communication links with neighbors in the communication network, as well as to detect link breakage. Once all nodes have transmitted a beacon, each node is provided with current neighborhood information in the communication graph.

Periodic broadcasting of beacons has several drawbacks, which are listed and discussed in detail by Heissenbüttel et al. in [12]. In general, beaconing is a proactive process and is performed independently of actual needs. Even if no task is to be completed, nodes periodically update their neighborhood tables. Heissenbüttel et al. [12] list the following direct and indirect impacts.

- Additional energy is consumed in order to transmit, receive, and process beacons.
- Beacons interfere with regular data transmissions and therefore, increase the number of collisions, which causes additional retransmissions.
- Beaconing introduces additional control overhead. The bandwidth is partially used for control traffic and is not available for user data.
- Due to fluctuations in the wireless channels and node mobility, the topology changes frequently and therefore a node's neighborhood table is likely to be inaccurate. Links

may appear or disappear. Until the next update, the neighborhood information of a node is outdated and may cause avoidable retransmissions, suboptimal application layer decisions, or even failure of the application itself.

Füßler et al. [17] remark that sending beacons at a higher rate can lower the aforementioned inaccuracies, but at the cost of a higher load on the bandwidth and increased energy consumption.

Especially in dense networks and networks that are highly mobile, e.g., networks consisting of mobile nodes that move at higher speed, beaconing often fails to provide a node with current and correct neighborhood information. However, for applications like localized geographic routing, the quality of the routing decisions heavily depends on appropriate neighborhood information.

Sanchez et al. [22] state that the effects of the aforementioned problems on real-world deployments should not be underestimated. This has been confirmed by several studies [12, 157, 158].

Beaconless algorithms (also called *contention-based* or *receiver-based algorithms*) are designed to avoid problems arising from the use of beaconing. They are fully reactive and work without maintenance of neighborhood tables [18]. In this context, the term *reactive* means that the algorithm is executed on demand only and must not involve any proactive component. Moreover, beaconless algorithms assume that a node is completely unaware of its network neighborhood at algorithm start.

Although beaconless algorithms are used to solve different algorithmic problems and vary a lot in their specific details, descriptions, and underlying assumptions, the *archetype* or *generic scheme* of any beaconless algorithm can be described as follows.

***Generic scheme* executed by a node u in order to solve problem P:**

1. At algorithm start, node u is unaware of its network neighborhood. It is possibly provided with some initial node knowledge, such as its geographic position, or node ID. Node u locally broadcasts a *request*, possibly containing (parts of) its initial knowledge. Then it starts a delay timer of duration t_{max} (a fixed time slice which is either globally known to all nodes, or which is included in the *request*), and waits for *responses* in the meantime.

2. The *request* from u is overheard by all network neighbors v of u. On reception, all nodes $v \in N_1(u)$ inspect the *request* synchronously.
   - If a node v decides, based on its initial node information and the information contained by the *request*, that it is not of any help to solve problem P, then it decides to stay passive for the moment (i.e., performs no operation), but keeps on listening to messages sent by other nodes. Overhearing further responses may lead to its reactivation.
   - Otherwise, if v considers itself important for solving problem P, it schedules sending a *response*, possibly containing (parts of) its initial node knowledge, after a delay whose duration is a fraction of t_{max}. The duration of the delay

timer is computed based on a function that usually depends on the node's initial knowledge and possibly on information contained in the *request*. It is designed so that nodes that are more appropriate for solving problem P answer earlier than those nodes that are less appropriate.

3. While a neighbor v of u waits for time-out of its delay timer it listens to other messages.

   - In case such a node v overhears a response by some other node w, which is more appropriate to solve problem P or obviates the need for sending the *response*, then v decides to become passive, cancels sending of its scheduled *response*, but keeps on listening to messages sent by other nodes. Overhearing further responses may lead to its reactivation.
   - Otherwise, if it has either not overheard any other *response* or decides to still be important for solving P, it locally broadcasts its response upon time-out of the delay timer anyway.

4. While the delay timer of node u has not timed-out, it is ready to receive and process *responses*. Typically, after having received the first response or a fixed number of responses, node u is provided with enough information to solve problem P. In this case, it may immediately transmit a *stop* message, which causes all nodes in $\mathrm{N}_1(u)$ to cancel their running delay timers and cancel transmission of scheduled messages.

   After time at most $\mathrm{t}_{\max}$, node u has received at least 0 and at most $|\mathrm{N}_1(u) - 1|$ many *responses*. The former case translates to a situation where all neighboring nodes consider themselves irrelevant for solving problem P. The latter case typically does not arise, except for worst-case situations, since it is equivalent to requesting a single beacon message from all neighboring nodes in $\mathrm{N}_1(u)$. In regular cases, node u receives one or several *responses*. Based on this information, node u either solves problem P directly, which is the typical case, or may initiate another round of algorithm execution by sending another request, possibly now containing additional information collected from those nodes that have previously responded.

Nodes that decide to be passive in the above description may be of further use. In some algorithms, the fact that some nodes become passive and do not transmit a message is problematic, as they remain hidden (invisible) for others. This may cause some nodes to respond although their response is superfluous or even adverse. Therefore, some beaconless algorithms let these passive nodes further overhear the message exchange. If they witness a response that can be considered adversarial given the local knowledge of this passive node, then it can correct the outcome by sending an additional message, often referred to as *protest message*, e.g., in [18].

The benefit of a beaconless algorithm compared to its beacon-based counterpart is that the executing node computes the same output as if it was provided with full neighborhood knowledge. During the execution of a beaconless algorithm by some node u, only a few messages are transmitted, typically a constant number. Any beacon-based approach

would require transmission of at least $|\mathrm{N}_1(u)|$ many messages, which is in the order of $\mathcal{O}(n)$ in the worst-case. Hence, beaconless algorithm can actually help to resolve the problems arising with beaconing listed above: In total, fewer messages are being transmitted and energy resources are saved. It is also less likely that regular transmissions are being interfered and the overall control overhead is reduced. Additionally, the construction of partial neighborhood knowledge requires potentially less time and therefore, yields more current information and is less likely to be outdated.

These arguments are the prototypical motivation for use of beaconless instead of beacon-based algorithms.

Note that beaconless algorithms are not only advantageous regarding theoretical worst-case situations. Their practical implementability and advantages with respect to several metrics have been shown empirically in various simulation studies (see e.g., [18–22], just to name a few) as well as in actual testbed experiments (see e.g., [23–25]).

4.2 Greedy Routing

Beaconless Greedy routing algorithms shift the routing decision—which neighbor is the locally optimal choice for forwarding—to the neighbors by means of a *contention mechanism*. The numerous approaches can be categorized into two main groups: *aggressive* and *non-aggressive contention schemes*.[1].

Aggressive contention schemes usually make use of implicit acknowledgements. Staying in the terminology of the generic scheme described above, the initiating node u includes the packet that has to be forwarded in its initial *request*. The neighbor v which is the most suitable for forwarding has the shortest time-out and immediately broadcasts the next *request*, again including the packet. Node u then considers the forwarding by v as an implicit acknowledgement.

In contrast, *non-aggressive contention schemes* usually make use of explicit acknowledgements. Node u sends its *request*. The node v which is the most suitable for forwarding answers first with a *response*; this can be considered as an acknowledgement. Upon reception of the *response*, node u then forwards the actual data packet to v.

In the following, both groups are handled independently. Within a group, approaches are clustered whenever possible with regard to their objectives. To simplify and harmonize the description of the individual approaches, some terms have to be defined first (see Figure 4.1 for an illustration).

In the following, denote by s always the *source node* of a unicast packet which is destined at *destination* node d. The *forwarder*, or *forwarding node*, which currently holds the packet, is always denoted by v. A *candidate* for forwarding is denoted by c. The area of intersection of the transmission radius R of forwarder v and the circle centered at d with radius $\|dv\|$ is the *positive progress area* (PPA). It is represented by the shaded area in Figure 4.1. The line segment $\overline{sd}$ between the source and the destination is referred to as the *source-to-destination line*, or simply *sd*-line. Define the *forwarder-to-destination* and *candidate-to-destination lines* accordingly. The interior angle $\alpha = \measuredangle vcd$ for candidate

[1]To the best of the author's knowledge, this categorization was introduced by Dinh et al. [159]

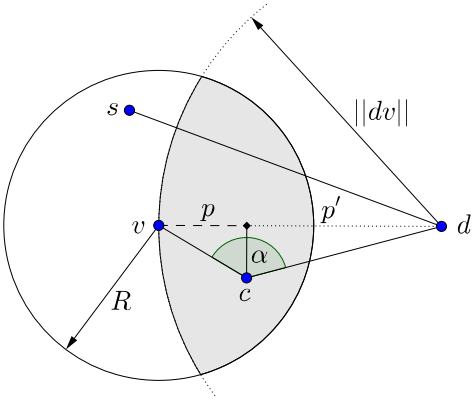


Figure 4.1: Illustrations of several notions in the context of beaconless Greedy routing. The shaded area represents the *positive progress area* (PPA) w.r.t. forwarder v and destination d. The dashed line segment p represents a candidate's *projected progress* w.r.t. the forwarder-to-destination line $\overline{vd}$.

c is used to express closeness to the vd-line. The *projected progress* of candidate c w.r.t. v and d is the length of the line segment p, resulting from projecting the position of c onto the vd-line. Line segment p' is the difference of $\|vd\|$ and p.

Aggressive contention algorithms often make use of so called *forwarding areas*. Typically, these are subareas of the positive progress area and ensure that any two nodes which are contained in the forwarding area can mutually overhear their messages. Only nodes that are located within the forwarding area participate in the contention process. If one candidate forwards the packet, then this is overheard by all other candidates, which then cancel their scheduled transmission. Figure 4.2 illustrates the most common forwarding areas in the shape of a (a) *sector*, (b) *circle*, and (c) *Releaux triangle*.

Finally, some of the approaches that are described below are so called *cross-layered approaches*. Cross-layer design means that parameters of two or more protocol stack layers can be retrieved and/or changed in order to achieve an optimization objective [160].

Aggressive Contention

The generic scheme underlying any aggressive contention Greedy routing approach can be summarized as follows.

***Generic aggressive contention scheme* executed by forwarding node v:**

1. Forwarder v locally broadcasts the packet, which includes its and the destination's position in the packet header.

2. Nodes in $\mathrm{N}_1(v)$ overhear this packet. Neighbors that find themselves not suitable for forwarding drop the packet, whereas all remaining neighbors start a delay timer. The actual delay depends on the optimization criterion. In general, the more suitable a node is for forwarding, the shorter is its delay.

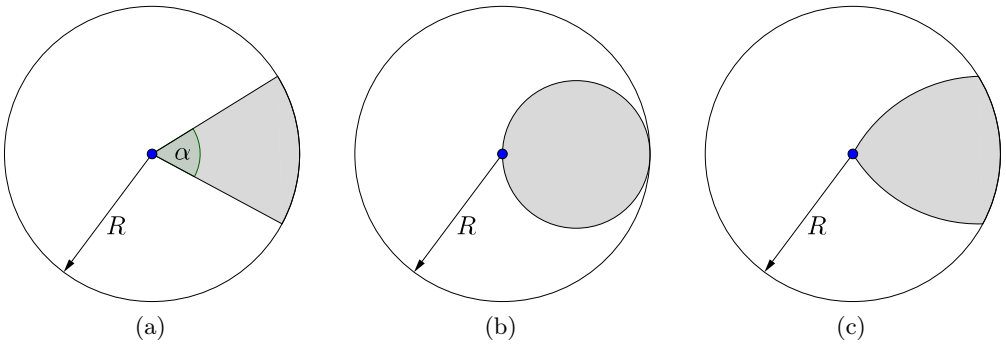


Figure 4.2: Illustration of different forwarding areas which are used in beaconless geographic Greedy routing algorithms: (a) *forwarding sector* with angle α, (b) *forwarding circle*, and (c) *forwarding Releaux triangle*.

.

3. If upon expiry of a node's delay timer, this node has not overheard a resubmission of the data packet, it updates the packet header and locally broadcasts it to its neighbors. Otherwise, it drops the packet.

Progress-based forwarding *Beaconless routing* (BLR) introduced by Heissenüttel et al. [16, 161] comes with three delay functions. The first basically implements the idea of *most forward within radius* (MFR) [162], which locally optimizes the progress. The delay of a candidate is inversely proportional to its progress and hence, the neighbor providing the largest progress forwards the packet. The second delay function implements a modified version of *nearest with forward progress* (NFP) [163]. The node with the least progress within the forwarding area (sector, circle, or Releaux triangle) has the shortest delay and forwards the packet, which increases the number of possible simultaneous transmissions in the network. The third delay function essentially favors those nodes that make large progress and are close to the *vd*-line at the same time. For the case of local minimum situations, two approaches are proposed: the local minimum node performs one-hop beaconing, extracts its adjacency in a planar subgraph, and uses the right-hand rule for recovery; or it makes use of *clockwise relaying* (CR), which is described in detail in Section 4.3.

Chen et al. [164] propose an extension of BLR [16, 161], henceforth referred to as *BLR with history* (BLRwH), where a node's delay is additionally influenced by recently overheard network traffic. Their assumption is that nodes which have recently overheard more network traffic are located in denser network areas than those having overheard little or no traffic at all; therefore, they are more suitable for relaying the packet. On the other hand, if the traffic load in the vicinity of a node is too high, then the delay should be penalized in order to avoid collisions. They suggest a delay function based on the progress criterion of BLR, which in addition accounts for a node's historic traffic

load. Recovery situations are resolved by employing beacon-based recovery strategies like GFG [36].

Endo et al. [165] introduce another modification of BLR, called *Distance-Aware Forwarding* (DAF). They propose to alter the delay function so that the farthest node within the forwarding area (sector) is assigned the shortest delay. In their analysis they show that this reduces the total delay in a multi-hop transmission at the cost of increased hop-length of the routing path.

Algorithm *improved progress position based beaconless routing* (IPBBLR) by Cao and Xie [166] is a minor modification of BLR. The delay of candidate c is computed as in BLR and weighted by $\cos(\measuredangle cvd)$ w.r.t forwarder v destination d in order to favor nodes that are closer to the forwarder-to-destination line. Their simulations show minor improvements in the delivery ratio.

Watanabe and Higaki propose *cooperated no-beacon geographic distance routing* (CNB-GEDIR) [167] which is in fact a beaconless route discovery scheme. However, if the route request messages used in this scheme are replaced with the actual data packet, it can be considered a classical aggressive contention Greedy routing algorithm. In fact it is then similar to BLR, since nodes providing large progress are favored during the contention.

Contention-based forwarding (CBF), introduced by Füßler et al. in [17,168], is similar to BLR [161], although proposed independently of it. CBF is a forwarding scheme composed of two parts: the *selection* of the next-hop is performed by means of contention, while *suppression* is used to reduce the number of packet duplications. CBF comes in three variants: *basic suppression*, *area-based suppression*, and *active selection*. The first two variants are aggressive contention algorithms, whereas the latter one belongs to the class of non-aggressive contention algorithms and is discussed later. In CBF, a node's delay is inversely proportional to the progress it provides towards the destination. In CBF with *basic-suppression* (henceforth CBF-BS), the forwarder initiates the contention process among all neighbors in the positive progress area, which may lead to packet duplication as nodes in the positive progress area may not be able to overhear the forwarding of the data packet. Therefore, in CBF with *area-based suppression* (henceforth CBF-ABS), only nodes in the forwarding Releaux triangle (see Figure 4.2c) participate in the contention process. Hence, CBS-ABS is in fact identical to BLR [161].

In *contention-based beaconless geographic routing* (CBGR) by Shi and Liu [169] it is considered how far the next hop can forward the packet. The candidate within the $\pi/3$-forwarding sector whose communication radius minimizes the Euclidean distance to the destination is assigned the shortest delay and forwards the packet. In static networks, nodes remember previous forwarding decisions and avoid repeated contentions. In mobile ad hoc networks, previous routing decisions are deleted after some fixed time interval and the contention mechanism is repeated. If no suitable next-hop exists within the forwarding area, the node performs beaconing and chooses another node within the positive progress area. To avoid routing loops, nodes drop a packet as soon as it would be forwarded to the same neighbor twice.

Blind geographic routing (BGR), introduced by Witt and Turau [21,170], addresses the problem which arises if more than one node forwards the packet simultaneously during the contention process. It also supports different delivery semantics (e.g., if the

destination's position is not known in advance or if several nodes in the vicinity of a destination position are targeted). BGR supports three-dimensional geographic positions. In principle, BGR is an algorithm framework and supports various forwarding areas and delay functions. The instance they use for simulations uses a forwarding sector and a delay function that favors nodes which are close to the destination. For handling cases in which two or more nodes (nearly) simultaneously forward a packet, the number of hops that a packet has been forwarded is stored in the packet header. In case a node has started a timer for a particular packet having a hop-count of k, the node only cancels its delay timer if it overhears forwarding of a packet with a hop-count larger than k. If the node overhears forwarding of a packet with a hop-count equals k, this packet stems from a simultaneous transmission and is simply ignored. BGR proposes two recovery strategies. If a circular forwarding area is used, it is turned by $\pi/3$ into a random direction (possibly multiple times) and the protocol is restarted with the rotated forwarding area. In case of a forwarding sector, it is augmented by $\pi/6$ in each step until either a suitable next-hop is detected, or the entire half-circle facing the destination has been covered.

Lee et al. propose *angled relaying with backoff time and (relay) cancellation* (ABC) [171], which uses a sector as the forwarding area. The sector is divided into equally spaced stripes (annuli) and each stripe is assigned a non-overlapping contention window. Stripes closer to the destination are assigned earlier contention windows. Each contention window is slotted into k time slots. Based on its own and the destination's position, a node can compute to which stripe it belongs. Any two nodes in the forwarding area which are located in the same stripe are assigned to the same contention window. Moreover, each node chooes one of the respective k time slots uniformly at random. The number of stripes and slots determines the collision probability. In case the forwarding sector is empty, the next forwarding sector in counter-clockwise direction is used. This event causes a *boundary-detection* process. Essentially the packet is routed around the void region until a node with a next-hop residing in the original forwarding sector is found. To solve future recovery situations, another control packet is routed further around the void region until the initial node on the void is reached. From then on, the void is detected and recovery is performed around the void using the known boundary nodes. However, since arbitrary edges are used for recovery, routing loops are possible.

Pizza Forwarding (PF) by Amadou and Valois [172] supports asymmetric communication links and non-isotropic radio propagation. Moreover, nodes do not have to know their communication range for delay timer computation. The current forwarder v partitions the plane into eight equal circular sectors (pizza slices) centered at v, see Figure 4.3. The four sectors facing the destination represent the Greedy forwarding area, whereas the two adjacent sectors facing the backwards direction are used for Recovery routing. These sectors are assigned priorities in $\{p_1, p_2, p_3\}$. The delay function of candidates c with highest and second highest priority p_3 and p_2, respectively, is similar to the third delay function of BLR [161] (see above) and favors those nodes that provide large progress and simultaneously minimize the angle $\measuredangle cvd$. Nodes add a (not further specified) random value to their delay timers in order to avoid collisions. The forwarder reacts by sending a SELECT packet upon relaying of the packet in order to avoid packet duplications. However, under the assumption that links are not necessarily bidirectional, this protocol

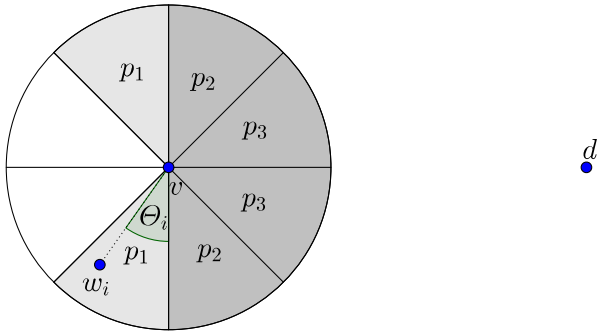


Figure 4.3: Illustration of the partitioning of the forwarding area (priorities p_2 and p_3) and the recovery area (priority p_1) as used in algorithms PF [172] and PFMAC [173]. Areas have different priorities p_i, v is the current forwarder, and d is the packet's destination. In recovery areas the delay of a node w_i also depends on the angle Θ_i. This figure is partially similar to [173, Figure 2]

.

still may produce packet duplications in case the forwarder does not overhear packet forwarding by the next-hop. The delay functions used in Recovery mode for handling local minima are discussed in Section 4.3.

Pizza-Forwarding Medium Access Control (PFMAC) [173] is a MAC-Routing cross layered version of Pizza-Forwarding (PF) [172]. Nodes perform asynchronous duty cycling. If a forwarding node wants to forward its data, it starts with a preamble transmission which is at least as long as the sleeping time of nodes. The preamble encodes information of whether greedy or Recovery mode is active. Nodes in the appropriate forwarding sectors that wake up and detect the channel activity start the PF [172] contention process, whereas nodes in other areas go back to sleep. In contrast to PF, the delay functions do not make use of a randomly chosen offset. Simulations show that PFMAC outperforms BOSS [174] (see non-aggressive contention algorithms below) in terms of energy consumption, end-to-end delay, and control message overhead.

Balancing of residual energy levels Jain et al. introduce *Energy aware Beacon-less Geographic Routing* (EABGR) [175]. It is explicitly designed for operation in three-dimensional networks. The forwarding region is a cone (3D sector). The candidate which minimizes the ratio of the quadratic progress to its residual energy level is assigned the shortest delay. That is, not only the closeness of a node to the destination, but also the state of its energy resources are taken into account. In case the forwarding volume is empty, it is shifted left or right, and the protocol is restarted.

In a follow-up study, Mishra and Gore [176] compare different combinations of delay functions and forwarding volumes (elliptic, parabolic, and regular cones) in EABGR. They propose to weight the delay function which is used in EABGR additionally by any of the following three factors: (1) angle $\alpha = \measuredangle vcd$, (2) the slope of c w.r.t. the forwarder-to-destination line, or (3) the corresponding angle $\alpha = \measuredangle cvd$. Their simulations

suggest that option (3) is best suited to differentiate between candidate nodes which have similar projected distances. Regular cones turn out to be best suited for dense networks, whereas elliptic cones are preferable in sparse and moderately-dense networks.

Base station assisted routing The *beaconless multihop routing algorithm* (BMR) by Nawaz et al. [177] is designed for static wireless sensor networks where the data sink is powerful base station outside the actual network. In the *network configuration phase* the base station transmits increasing zone-IDs while increasing transmission power. A node belongs to the smallest zone-ID it has overheard. This way the network is clustered into zones $Z_0, ..., Z_k$. In the *data communication phase* a forwarder locally broadcasts its packet including its zone-ID. Only nodes with smaller zone-ID start a delay timer. The latter is based on the nodes' residual energy and the nodes' zone-ID. Effectively, the node with the smallest zone-ID and the highest residual energy is the first to forward the packet. However, the forwarding area is not specified and hence, packet duplications may occur.

Jurdak et al. [178] present an approach which is similar to BMR, called *Directed Broadcast with Overhearing* (DBO). In a *setup phase*, a data sink broadcasts a packet over the entire network. All nodes at hop distance k from destination d are grouped into the same hop-zone. Once a node at hop distance k wants to send a packet to the destination, it locally broadcasts it and only nodes belonging to group $k-1$ start a random delay. One node relays the packet and other nodes in this hop-zone which overhear the relay transmission cancel their delay timers. As in the case of BMR, this may lead to packet duplications. DBO is a MAC cross-layer protocol. It combines directed broadcast at the network layer with CSMA and packet overhearing at the MAC layer.

Real-time routing The *receiver-based beaconless real-time routing protocol* (RBRR) by Yim et al. [179] enables a real-time communication service. The forwarder includes information about the average single hop delay (average delay between entering time to output queue and sending time of the last bit for a packet) in the packet header. Once the delay timer of a neighboring node expires, it only relays the packet if the real-time constraints can still be met with regards to the average single hop delay. The forwarding area is a Releaux triangle which is partitioned into concentric stripes (centered at the destination). Stripes are assigned non-overlapping contention timers. The stripe closest to the destination has the shortest delay. The order of nodes within a stripe is determined by a randomly chosen offset.

Multipath Al-Otaibi et al. introduce *multipath routeless routing* (MRR) [180], a multipath routing protocol designed for highly mobile networks. The forwarding area, called *route broadcast virtual channel*, is a rectangle of unlimited length that is fixed and centered at the position of the source node s and whose width is 2α, where $R/2 < \alpha < R$ is a fraction of the unit disk graph radius R (see Figure 4.4 for an illustration). Specification of this forwarding area is part of the packet header and only nodes located within this area participate in the multi-hop forwarding process. The actual forwarder selection is

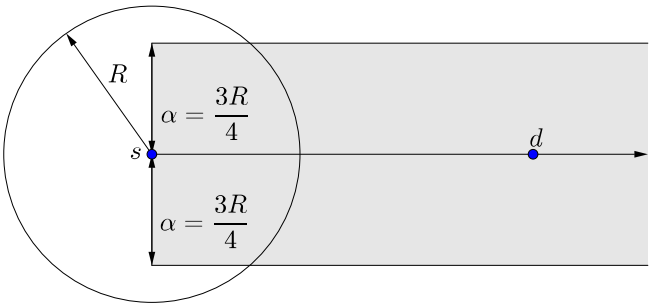


Figure 4.4: Illustration of the forwarding area used in algorithm MRR [180]. R denotes the unit transmission radius, s is the source and d the destination of the packet. Parameter $\alpha \in [R/2, R]$ specifies the width of the forwarding area. This figure is similar to [180, Figure 1]

.

similar to BLR [161], however, the delay function is based on three criteria: the receiver sensitivity, the distance of the candidate to the source-to-destination line, and the node's residual energy. The delay function is designed so that nodes receiving weaker signals, located closer to the boundaries of the forwarding area, and having more remaining energy have shorter delays [180]. Since nodes which are close to the border are preferred and the width of the forwarding area is larger than R, at least two non-interfering routes exist.

Multimedia streaming The *Link quality and Geographical beaconless Opportunistic Routing* protocol LinGO by Rosario et al. [181] is a cross-layered approach for enabling efficient video dissemination for mobile multimedia *Internet of Things*. Essentially, they make use of the standard aggressive contention principle to establish on demand a route from a source to a destination node, which is then used for forwarding the entire data stream. During the contention process, which is similar to MRR [180], all nodes in the positive progress area participate. The delay function regards a candidate's progress, its residual energy, as well as the link quality (based on *Signal-to-Noise Ratio* (SNR), *Signal-to-Noise plus Interference Ratio* (SINR), or the *Link Quality Indicator* (LQI))[2]. If the forwarder does not detect the relaying of the packet, or if the forwarding area is empty, it retries until a next-hop is found.

Vehicular networks *Vehicular ad hoc networks*, so called VANETs, comprise vehicle-to-vehicle and vehicle-to-infrastructure communications based on wireless local area network technologies [182]. Unlike in typical ad hoc networks, in VANETs it is typically assumed that nodes are provided with street maps and are possibly able to communicate with some fixed roadside infrastructure providing nodes with information on traffic flow, etc.

In [183], Füßler et al. propose *contention-based forwarding in street scenarios* (CBFSS), which adopts the *basic suppression* version of their protocol CBF [17] for use in VANET scenarios. The forwarding node locally broadcasts the packet and all nodes in the positive

[2]For details on these metrics, the reader is referred to the textbook by Karl and Willig [3].

progress area start their delay timers inversely proportional to their progress towards the destination. The node with earliest timeout performs the next forwarding step. The authors claim that packet duplications rarely occur in street scenarios since cars can only be located in narrow areas (streets) and therefore no further suppression scheme is required. Moreover, in order to avoid packet duplications, nodes store the unique packet IDs and simply drop duplicated packets.

The *Routing Protocol with Beacon-Less* (RPBL) by Sasaki et al. [184] is similar to CBFSS [183]. Nodes are provided with road maps and can locally compute the shortest path to the destination. Instead of selecting candidates that maximize progress towards the destination, the candidate maximizing the progress towards the next intersection on the shortest path (called *temporary destination*) is determined in the contention process.

Taxonomy Table 4.1 lists and summarizes the aforementioned aggressive contention, geographic routing algorithms in chronological order.[3] For each algorithm, this table lists the following:

k-locality refers to the degree of k-locality required by nodes to compute the next-hop.
Path strategy specifies if it is single path or multi path approach.
Metrics specifies which metrics are considered for selection of the next-hop during the contention process.
Stateless is checked with '✓', if the algorithm is stateless.
Cross layer is checked with '✓', if the algorithm explicitly combines multiple network layers.
3D is checked with '✓', if the algorithm explicitly handles 3D node positions.
VANET is checked with '✓', if the algorithm is explicitly designed for application in vehicular ad hoc networking scenarios.
Reference refers to source(s).
Year lists the earliest point in time of publication.

[3] The only work not considered in this table is [176], as it does not describe an individual algorithm, but rather compares different combinations of delay functions and forwarding areas on top of another algorithm.

Algorithm	k-loc.	Path strategy	Metrics	State-less	Cross layer	3D	VANET	Ref.	Year
CBF-(A)BS	1	Single	D	✓				[17, 168]	2003
CBFSS	1	Single	D	✓			✓	[183]	2004
BLR	1	Single	D	✓				[16, 161]	2004
IPBBLR	1	Single	D	✓				[166]	2005
BGR	1	Single	D	✓[a]		✓		[21, 170]	2005
CNB-GEDIR	1	Single	D	✓				[167]	2007
BLRwH	2	Single	D,δ					[164]	2007
CBGR	1	Single	D					[169]	2008
DBO	1[b]	Multi	H		✓			[178]	2009
EABGR	1	Single	D,E	✓		✓		[175]	2009
DAF	1	Single	D	✓				[165]	2010
ABC	2	Single	D	✓				[171]	2010
PF	1 [c]	Single [d]	D	✓				[172]	2010
BMR	1[b]	Multi	D,E					[177]	2011
MRR	1	Multi	S,D,E	✓				[180]	2011
PFMAC	1 [c]	Single [d]	D	✓	✓			[173]	2011
RBRR	1	Single	D,T	✓				[179]	2013
RPBL	1	Single	D	✓			✓	[184]	2013
LinGO	1	Single	S,D,E	✓	✓			[181]	2014

[a] Only if nodes store a constant number of previously forwarded packets
[b] Requires a global setup phase
[c] Recovery mode requires 2-hop information
[d] Packet duplication may occur due to unidirectional links
Metrics: D=Distance, E=Energy, H=Hops, T=Time, S=Signal strength, δ=Node density

Table 4.1: Taxonomy of aggressive contention, reactive Greedy routing algorithms in chronological order.

Non-Aggressive Contention

The generic scheme underlying any non-aggressive contention, Greedy routing algorithm can be summarized as follows.

***Generic non-aggressive contention scheme* executed by forwarding node v:**

1. Forwarder v locally broadcasts a REQUEST, including its and the destination's position.
2. Nodes in $N_1(v)$ overhear this packet. Neighbors that find themselves not suitable for forwarding stay passive, whereas all remaining neighbors start a delay timer. The actual delay depends on the optimization criterion. In general, the more suitable a node is for forwarding, the shorter is its delay.
3. Upon expiry of a node's delay timer, it answers the REQUEST by sending a RESPONSE including its position.
4. Nodes whose delay timers have not yet expired and which overhear sending of a RESPONSE, cancel their delay timers and refrain from sending a RESPONSE.
5. Upon reception of a RESPONSE, forwarder v locally broadcasts a SELECT, which announces the next forwarding hop. Typically, this is the node which has answered first. Nodes overhearing this SELECT and that still have a running delay timer, cancel it and refrain from sending a RESPONSE.

In this general scheme, the actual DATA packet is either contained in the REQUEST, or in the SELECT.

In contrast to aggressive contention algorithms, in non-aggressive contention algorithms the forwarder's positive progress area (see Figure 4.1) can be used exhaustively. Other nodes in the progress area overhear the RESPONSE or the SELECT. In both cases, they cancel their delay timers and refrain from answering the REQUEST. While using the entire progress area, the delivery rate is maximized at the cost of additional message overhead, which in turn reduces the available bandwidth and causes higher energy consumption.

The above generic scheme resembles the RTS/CTS handshake as used in IEEE 802.11 [185], which is based on the MACAW protocol [186]. In MACAW, when node a wants to send some DATA to a node b, it first sends a *request-to-send* (RTS). If c has properly received the RTS, b answers with *clear-to-send* (CTS). If a has properly received the CTS, then it starts transmitting the actual DATA, which is then *acknowledged* with an ACK by b upon successful retrieval. Many of the non-aggressive beaconless geographic routing algorithms borrow this RTS-CTS-DATA terminology. It is therefore used interchangeably with the REQUEST-RESPONSE-SELECT terminology in the remainder of this work.

Variations of the generic approach *Contention-based forwarding* (CBF) with *active selection* (henceforth, CBF-AS), introduced by Füßler et al. [17, 168], is a one-to-one implementation of the generic non-aggressive contention scheme. First, the forwarding

node locally broadcasts the RTS including the data packet. A candidate's delay is inversely proportional to the progress it provides towards the destination. The candidate whose timer expires first answers with a CTS. The forwarder then locally broadcasts a SELECT indicating which node has been selected for forwarding. This suppresses all other timers of nodes in the positive progress area. In case of a local minimum situation, standard beacon-based approaches like FACE [36] may be used for recovery.

Chawla et al. [187, 188] propose *Guaranteed delivery beaconless forwarding* (GDBF). It proceeds just like CBF-AS: the delay functions are similar, but the data is contained in the SELECT rather than in the initial broadcast by the forwarder. In case there are two or more nodes at the same distance with respect to the destination, which causes a collision at the forwarder, the forwarder requests a resubmission, but this time with randomly chosen timer values. The mechanism used for recovery in local minimum situations is discussed in detail in Section 4.3.

Distributed Passive Routing Decisions (DPRD) introduced by Škraba et al. [189, 190] is a CSMA/CA based cross-layered version of the generic scheme. It assumes that nodes are able to do carrier sensing. The main emphasis of this work is the analysis of two classes of delay functions which are based on different physical layer parameters. Their analysis suggests the following: When the delay function is modeled as an exponential function, then it is possible to optimize both delay and probability of collision, even at high node densities. However, when the delay functions is modeled as a linear function, then the collision probability can reach unacceptable levels at high node densities.

Beacon-less on demand strategy for geographic routing in wireless sensor networks (BOSS), proposed by Sanchez et al. in [23, 174], is designed for operation in error-prone networks. Data is transmitted as part of the initial REQUEST by the forwarder, as in the case of CBF-AS [17, 168]. Only nodes that are located in the positive progress area and that have correctly received the packet are allowed to participate in the contention process. The positive progress area is sub-divided into stripes (see Figure 4.5). Stripes are assigned unique and non-overlapping delay intervals. Nodes within a particular delay zone add a random value to the zone's minimum delay value such that the overall delay remains within the zone's assigned delay interval. In local minimum situations BOSS makes use of the recovery mechanism of GDBF [187, 188] or the dedicated recovery algorithm Angular Relaying (AR) [18, 107], both of which are described in Section 4.3.

Aguilar et al. [192] propose a beaconless geographic cross-layer protocol, called *CoopGeo*, for cooperative wireless ad hoc networks. This combines geographic routing and geographic relay selection. CoopGeo is composed of several components: the MAC-network cross-layer protocol *beaconless greedy forwarding* (BLGF), based on BOSS [174]; the MAC-network cross-layer protocol *beaconless recovery forwarding* (BLRF), based on BFP [18] (see Section 4.3); and a beaconless MAC-PHY cross-layered relay selection scheme, which is executed if the data packet cannot be encoded correctly at the selected forwarder. BLGF is similar to BOSS. However, all nodes in the positive progress area participate in the contention process, independent of whether the data (contained by the REQUEST) could be decoded correctly or not. Upon expiry of the delay timer, a node transmits its RESPONSE and indicates in this packet if relay cooperation is needed due to error decoding. In case relay cooperation is required, all nodes in a particular area (the Releaux

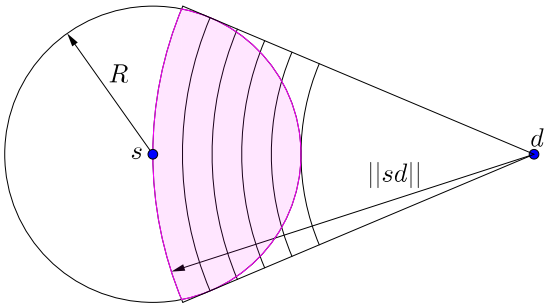


Figure 4.5: Illustration of the partitioning of the positive progress area (pink) as used in algorithms GeRaF [27, 191] and BOSS [174]. R denotes the unit transmission radius, s is the source and d the destination of the packet. This figure is similar to [174, Figure 3]

.

triangle or the RNG-lune between the forwarder and the selected next-hop) which have received and correctly decoded the data start another timer-based contention, where the timers are based on cooperative link quality in terms of average point-to-point *symbol error rate* (SER). The first node whose timer expires becomes the relay node and relays the data to the selected next-hop, which then uses maximum ratio combining for data decoding. CoopGeo uses BFP [18, 107] (described in Section 4.5) for handling local minimum situations.

Balancing residual energy levels *Implicit geographic forwarding* (IGF) by Blum et al. [19] is a MAC-cross layer approach which modifies the 802.11 MAC protocol for contention-based selection of the next forwarding hop. At first, this protocol proceeds just like the generic scheme formulated above. A node's delay timer depends on its progress towards the destination, its residual energy level, and an additional random value. After the RESPONSE has been received by the forwarder, communication proceeds as in IEEE 802.11 semantics (CTS-DATA-ACK). To avoid collisions, IGF uses a $\pi/3$-forwarding sector. If no suitable next-hop can be found, the forwarding area is shifted and the protocol is restarted.

State-free Implicit Forwarding (SIF) [193] and *On-demand Geographic Forwarding* (OGF) [20], both proposed by Chen et al., are similar to IGF [19]. Instead of using only a forwarding sector, both make use of the entire positive progress area. To avoid collisions, nodes are assumed to be able to perform carrier sensing within at least two times the regular transmission range. The delay functions used in SIF and OGF are also similar to the one proposed in IGF. However, OGF differs in so far as nodes maintain routing tables in which previous forwarding decisions are stored. A contention-based next-hop forwarding decision is only performed if the routing table does not contain an appropriate entry. For handling local minimum situations, SIF proposes to gradually increase transmission power until a suitable next-hop is detected, or to make retries at a later point in time. In OGF, Partial Source Routing (PSR) is proposed to handle voids. It

is essentially a partial k-hop flooding technique for route discovery. Once the destination node or a node being closer to the destination than the local minimum receives the packet flooded, it uses path information stored in the packet to answer the minimum node. It thereby establishes a route to the destination.

Optimizing energy efficiency Algorithms *EBGR* [194] (also called *EEGR* in [195]) and *MEGAN* [196] are designed for energy efficient routing. The idea underlying all of these protocols is as follows: The current forwarder computes first the ideal position of the next-hop with regards to energy consumption. Then, the forwarder starts a conventional contention process where nodes close to the ideal position have shorter delays and are favored for forwarding.

Energy-efficient Beaconless Geographic Routing (EBGR) by Zhang and Shen [194] is a follow-up and further development of their algorithm *Energy-Efficient Geographic Routing* (EEGR) introduced in [195]. In EBGR, the ideal next-hop position p (a point on the forwarder-to-destination line) is calculated based on the optimal forwarding distance in terms of minimizing the total energy consumption for delivering a packet from the current forwarder v to destination d [194]. The circle centered at p with radius at most $\|vp\|$, called *relay search region*, is then subdivided into concentric annuli of equal covered area. Like in BOSS [174], each annulus is assigned a unique and non-overlapping time period. The closer the annulus is to the optimal position p, the shorter is the delay. Moreover, the delay function used in EBGR ensures that two nodes that are located in the same annulus have different timeouts. For recovery from local minimum situations, EBGR uses the dedicated recovery algorithm AR [18, 107] (see Section 4.3).

Mitton et al. introduce in [196] *Mobility assisted Energy efficient Georouting in energy harvesting Actuator and sensor Networks* (MEGAN). It aims at prolonging the overall network lifetime, rather than reducing the energy consumption over a single path [196]. It assumes that nodes are mobile and have the capability to harvest energy. The current forwarder v of a packet first estimates the ideal position p based on its current energy consumption and harvesting rates, as well as its maximal and current residual energy levels. Then, the forwarder v adjusts its transmission range to $\|vp\| + \epsilon$ and locally broadcasts the REQUEST including the positions of p and itself. Nodes overhearing this transmission start a delay timer so that the node with more residual energy after the movement to ideal position p is the first node that answers the REQUEST. This node then moves towards p. If it finds a good spot for energy harvesting on its way, it stops and proceeds with the next routing step.

The author of this thesis is well aware of algorithm *energy-efficient beaconless geographic routing with energy supply* (EBGRES) by Jumira et al. [197]. However, the contents and results presented there are suspiciously similar to those presented first in EBGR [194]. Therefore, this work is simply disregarded.

Duty cycling *Geographic Random Forwarding* (GeRaF) by Zorzi and Rao [27, 191] considers problems arising with duty-cycling. The idea is to use whatever node is available at any given time, without waiting for a specific node to wake up [198]. In GeRaF the

positive progress area is partitioned into several priority regions ordered by distance (see Figure 4.5). Each region is assigned a unique time slot upon sending of the REQUEST by the forwarder. Nodes residing in the region with the highest priority are the first to answer and their RESPONSE cancels transmissions by nodes in lower priority regions. If there are several nodes within the region with highest priority, all of them transmit their RESPONSE simultaneously, which causes a collision at the forwarder. This is detected and a collision resolution algorithm is executed. Nodes in the highest-priority region retransmit their RESPONSE with probability 1/2 until exactly one node has successfully transmitted its packet. In case the forwarding area is empty, the protocol is restarted at a later point in time when another set of nodes is awake.

In a follow-up paper [198], Zorzi proposes a collision avoidance scheme for GeRaF. Traditional carrier sense schemes only partially avoid collisions if nodes wake up asynchronously and hence, hidden terminal problems may occur. To avoid this, nodes listen to the radio channel for a particular time frame T_{sens} before transmitting a packet. If within this time frame no activity has been sensed, the transmission can be started. Otherwise, the node backs off and retries transmission at a later point in time. In order to actually avoid collisions, the length of time frame T_{sens} is based on the relationship between the sensing time and the transmission schedule which is followed by active nodes.

The *cross-layer module* (XLM) by Akyildiz et al. [29] is similar to GeRaF, but it also incorporates transport layer functionalities. The protocol is based on a concept called *initiative determination*. Nodes only participate in the contention process if the REQUEST is sent over a channel with adequate quality (determined on the basis of the *Signal-to-Noise Ratio* (SNR)), if the node is not congested, if enough buffering capacities are available, and if the residual energy allows for participation in the forwarding process. In an extension of this algorithm, called XLP [28], Vuran and Akyildiz propose to add a sweep line-based recovery strategy for handling local minimum situations. This is explained in detail in Section 4.3.

Taking network density into account *Priority-based stateless geo-routing* (PSGR) by Xu et al. [32, 199] also subdivides the positive progress area into unique forwarding zones. Unlike in GeRaF [191] or BOSS [174], however, the number of zones is not predetermined, but computed dynamically based on estimated node density. For an estimation of the node density, the forwarder overhears network traffic for a certain amount of time and counts the number ρ of distinct nodes to which it is connected. Then, the positive progress area is subdivided into stripes, such that each stripe covers area A/ρ, where A is the area covered by the positive progress area; i.e., assuming that nodes are uniformly distributed, each stripe contains one node in expectation. Each stripe is assigned a unique delay timeout and the first node to answer is the next-hop for forwarding. In order to further minimize the delay time, the authors introduce another heuristic, which first subdivides the positive progress area into forwarding zones, such that all nodes within a particular zone can overhear each other. Each zone is assigned a unique delay window. The formerly outlined approach is applied within each zone, i.e., each zone is further subdivided into

stripes containing one node in expectation. Their routine for handling local minima, called *Bypass*, is described in detail in Section 4.3.

Real-time routing The *contention-based beaconless real-time routing protocol for wireless sensor networks* (CBRR) by Huang and Wang [200] aims at energy efficient end-to-end real-time routing and is similar to the aggressive contention real-time routing protocol *RBRR* by Yim et al. [179] (see Section 4.2). Only those nodes that meet the packet's real-time constraints participate in the contention process. The actual delay function takes into account a combination of the candidate's progress, its residual energy, the number of packets waiting in its output queue, as well as a random value. A node stores previous forwarding decisions in a routing table. If a packet is to be forwarded, the node first checks its neighborhood table and only starts the contention process if no suitable neighbor could be determined. The authors also describe a two-hop version of the protocol, where nodes overhear transmissions of other nodes and update their two-hop neighborhood tables accordingly.

Variable transmission range Das et al. propose in [34] the partially beaconless algorithm *Geographic Routing with Variable transmission range* (GRoVar). It is assumed that nodes can vary their transmission range and that nodes perform beaconing only at low transmission range $\mathrm{R}_{\mathrm{init}}$, a fraction of the maximal transmission range $\mathrm{R}_{\mathrm{max}}$. If a forwarder v wants to forward a packet, it chooses among all neighbors within range $\mathrm{R}_{\mathrm{init}}$ the one that minimizes the cost w.r.t some metric such as distance to the target or power over progress [201]. If no such neighbor exists, the transmission range is increased stepwise. Then, the generic non-aggressive contention scheme is used to find a neighbor maximizing the progress towards the destination. In case that no appropriate next-hop can be found, even with a maximal transmission range $\mathrm{R}_{\mathrm{max}}$, the dedicated recovery algorithm AR [18] (see Section 4.3) is used.

The *integrated MAC/Routing protocol* (MACRO) by Galluccio et al. [202] aims at minimizing energy consumption by exploiting the nodes' capability to adjust their transmission ranges. It is assumed that nodes perform asynchronous duty cycling. If a forwarder v wants to forward a packet, it transmits several wake-up messages until all forwarding candidates are active. It then locally broadcasts a REQUEST. Nodes overhearing it start their delay timers depending on their *weighted progress* (the ratio of the distance progress towards the destination to the transmission power used by the forwarder). Upon timer expiration, a node answers by transmitting its weighted progress. The forwarder estimates, based on a probabilistic function, if an increase in transmission power will lead to a next-hop with higher weighted progress. If no higher weighted progress is expected, the packet is forwarded to the candidate currently maximizing the weighted progress; otherwise, the procedure is repeated with increased transmission power.

Multipath routing The *Beacon-less Geographic Multipath routing protocol* (BGM) by Dong et al. [203] aims for reliable geographic routing using maximally node-disjoint multiple routing paths. Similarly to the aggressive contention algorithm MRR [180], the

source node s partitions the area between s and destination d into k zones z_i, using elliptic curves instead of rectangles. The packet, including the information which is necessary for a node to determine in which zone it is located in, is then injected into each zone and forwarded independently. Let v be a forwarder in zone z_k. Upon transmission of a REQUEST, nodes in the positive progress area start their delay timers. The delay function favors those nodes that are in the same zone and provide large progress towards the destination. Nodes in neighboring zones that are also in the positive progress area are penalized and compute their delay based on the proximity to the respective boundary curve. This way packets are allowed to leave their zone, but keep close to the zone boundary for eventual return. In addition, delay timers are multiplied with a random number in order to avoid collisions. The node with earliest time-out wins the contention and is selected for forwarding.

Convergecast Zhang introduces in [26] a context-aware duty cycle assignment protocol called *Receiver-Based Heading* (RBH). It assumes that there are several data sources that periodically send data towards a common data sink. The sending period is globally known to all nodes and they are programed to wake up during packet generation at the sources. This is called the *detection phase*. This phase creates a tree, rooted at the destination and connecting it with all sources, using a standard non-aggressive contention process. Nodes closest to the destination in the positive progress area are assigned the shortest delay and data packets are routed from the sources to the destination. All forwarders along this path, called *heads*, build the desired tree. In the succeeding *heading phase*, all non-head nodes are set back to sleep mode to conserve energy.

Based on the regular packet generation interval and the transmission durations of packets in the detection phase, head nodes compute during which time frames they have to be awake in order to cover upcoming forwarding events. This protocol performs context-aware duty cycling in so far as the cycles are computed on demand and dependent on the actual node deployment.

Hybrid contention *Hybrid Contention-Based Geographic Routing* (HCGR) by Dinh et al. [159] combines aggressive and non-aggressive Greedy routing into a single protocol. The positive progress area of a forwarding node v is partitioned into an *aggressive contention area* (AA) and a *non-aggressive contention area* (NA) (see Figure 4.6a). The forwarder locally broadcasts the DATA packet. All nodes u in AA and NA (i.e., in the entire positive progress area) start their delay timers using the same delay function. The latter favors those nodes that are close to the vd-line, and which provide large progress. The authors claim that their delay function always assigns shorter delays to nodes in AA than to nodes in NA. It is, however, easy to construct counter-examples where this is not correct.[4] Nodes in AA immediately forward the DATA packet, whereas nodes in NA send a RESPONSE upon timer expiration. Upon reception of DATA or RESPONSE sent by node u, forwarder v acknowledges the transmission with a SELECTION containing

[4] For example, in Figure 4.6a the delay of node e is $0.15 \cdot \mathrm{t_{max}}$, whereas the delay of node f is $0.13 \cdot \mathrm{t_{max}}$ although it is outside of AA.

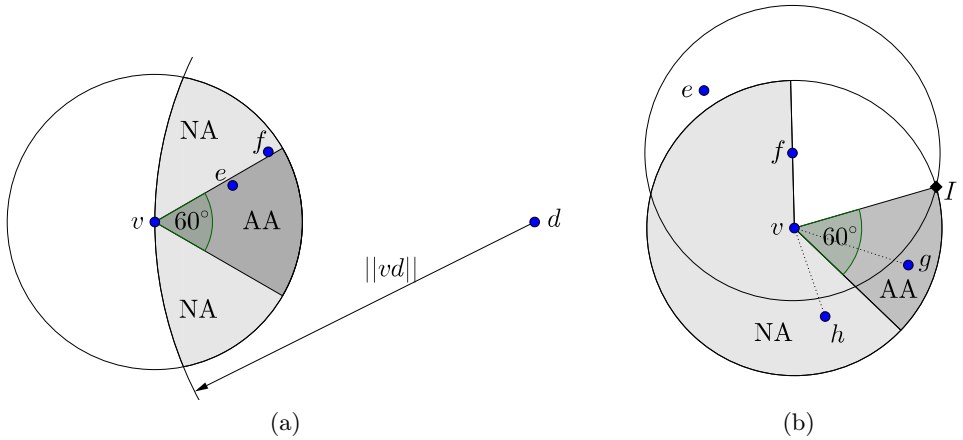


Figure 4.6: (a) Illustration of the partitioning of the positive forwarding area into the aggressive contention area (AA) and the non-aggressive contention area (NA) as used in algorithm HCGR [159]. In this example, delay of node e is larger than that of node f. (b) In Recovery mode the aggressive (AA) and non-aggressive (NA) contention areas are defined in dependence of the previous two hops e and f. In this example g minimizes the interior angle at v w.r.t. I and is selected for forwarding. Figures (a) and (b) are partially similar to [159, Figures 1 & 2]

.

the position of u in order to suppress timers of all other nodes. If forwarder v is a local minimum, a Recovery mode is applied (see Figure 4.6b). In Recovery mode the aggressive and non-aggressive contention areas are defined relative to the positions of the previous two hops e and f. The delay of a node g is proportional to the interior angle $\measuredangle Ivg$. The actual forwarding procedure is similar to Greedy mode. Since the edges used in Recovery mode are not necessarily free of edge intersections, the resulting routing path may loop.

Vehicular networks In [204] Ruiz et al. extend the non-aggressive Greedy forwarding algorithm BOSS [174] (see above) to vehicular networking scenarios and introduce the *Beacon-less Routing Algorithm for Vehicular Environments* (BRAVE). Nodes are provided with street maps and can locally compute the shortest path on the street map from the current position to the destination. A forwarding node v essentially executes algorithm BOSS [174]. In case of a crossing, the forwarding node starts the contention process among all nodes that are located in between the next and the after next crossings. In case of a local minimum, BRAVE employs a *store-carry-forward* approach, i.e., the node stores the packet until a suitable next-hop becomes available again.

Contention-based beaconless packet forwarding (CBBPF) by Asgari et al. [205] is also similar to BOSS [174] (see above). Nodes are not provided with street maps and simply try to forward the message as close to the destination as possible using the contention mechanism from BOSS [174]. In contrast to BOSS, the actual data is sent last as a part

of the SELECT. CBBPF is a cross-layered approach. The forwarder's MAC layer sends its data delivery report to the network layer to indicate whether the winning candidate successfully received the data or not [205]. In case of a local minimum, CBBPF also performs store-carry-forward.

Intelligent Beaconless geographical forwarding (IB) by Ghafoor et al. [206] is a modification of IEEE 802.11 RTS/CTS frames for packet forwarding in VANET scenarios. Nodes are provided with street maps and can identify if they are in between two intersections or on an intersection. The RTS sent by the forwarder includes information about whether the forwarder is in between or on an intersection. In the former case, all nodes overhearing the RTS start a delay timer. The delay function favors those nodes that are moving in the same direction of the forwarding node and that receive the RTS with a strong power level signal. In case the current forwarder is on an intersection, the delay function includes a *greediness factor* representing the closeness of the node to the destination, in addition to the direction and the signal power level.

In [207] Nzouonta et al. propose similar modifications of IEEE 802.11, but for application in a non-local route discovery scheme.

Security aspects For the sake of completeness, at this point it shall be mentioned that there are several papers [208–210] dealing with security aspects in beaconless forwarding algorithms, in particular in the context of vehicular networking. This line of research considers the problems arising with malicious nodes disturbing the contention/forwarding process. However, for the present work, these aspects are far out of scope.

Taxonomy of non-aggressive contention schemes Table 4.2 lists and summarizes the aforementioned non-aggressive contention geographic routing algorithms in chronological order. For each algorithm, this table lists the following:

k-locality refers to the degree of k-locality required by nodes to compute the next-hop.
Path strategy specifies if it is single path or multi path approach.
Metrics specifies which metrics are considered for selection of the next-hop during the contention process.
Stateless is checked with '✓', if the algorithm is stateless.
Cross layer is checked with '✓', if the algorithm explicitly combines multiple network layers.
VANET is checked with '✓', if the algorithm is explicitly designed for application in vehicular ad hoc networking scenarios.
Reference refers to source(s).
Year lists the earliest point in time of publication.

Algorithm	k-loc.	Path strategy	Metrics	State-less	Cross layer	VANET	Ref.	Year
IGF	1	Single	D,E	✓	✓		[19]	2003
GeRaF	1	Single	D	✓	✓		[191]	2003
DPRD	1	Single	D	✓	✓		[189, 190]	2004
PSGR	1	Single	D				[32, 199]	2004
SIF	1	Single	D,E	✓	✓		[193]	2005
XLM	1	Single	D	✓	✓		[29]	2006
GDBF	1	Single	D	✓			[187, 188]	2006
OGF	1	Single	D,E		✓		[20]	2007
MACRO	1	Single	E	✓	✓		[202]	2007
BOSS	1	Single	D,L	✓			[23, 174]	2007
GRoVar	1	Single	D,H,E	✓			[34]	2007
CBRR	1/2	Single	D,E,Q				[200]	2010
EBGR/EEGR	1	Single	E	✓			[194, 195]	2010
BRAVE	1	Single	D	✓		✓	[204]	2010
BLGF	1	Single	D	✓	✓		[192]	2011
HCGR	1	Single	D	✓			[159]	2011
BGM	1	Multi	D	✓			[203]	2012
RBH	1	Single	D	✓	✓		[26]	2012
IB	1	Single	D,S	✓	✓	✓	[206]	2013
MEGAN	1	Single	E	✓			[196]	2013
CBBPF	1	Single	D	✓	✓	✓	[205]	2013

Metrics: D=Distance, E=Energy, H=Hops, S=Signal strength, Q=Queue length, L=Link quality

Table 4.2: Taxonomy of non-aggressive contention, reactive Greedy routing algorithms in chronological order.

4.3 Recovery Routing

In the following, approaches to the *beaconless recovery problem* are reviewed. Some of these approaches, especially those that guarantee message delivery, are of particular interest for later chapters of this thesis. For this reason, these approaches are explained in more detail. The beaconless recovery problem can be formalized as follows:

Definition 4.1 (Beaconless recovery problem)**.** Let $G = (V, E)$ be a connected geometric graph and $u \in V$ a local minimum node (see Definition 2.7) w.r.t. destination node $d \in V$. The *beaconless recovery problem* refers to the algorithmic challenge of computing locally and hop-by-hop, a connected node sequence $S = \langle u = s_1, s_2, ..., s_k \rangle$ in G, such that S is a recovery path (see Definition 2.8) for escaping the local minimum at node u, without making use of beaconing, i.e., without intentionally gathering full neighborhood information.

The majority of beaconless recovery algorithms makes use of *sweep lines* and *sweep curves* for detection of the next hop on the recovery path. Their general idea can be summarized as follows.

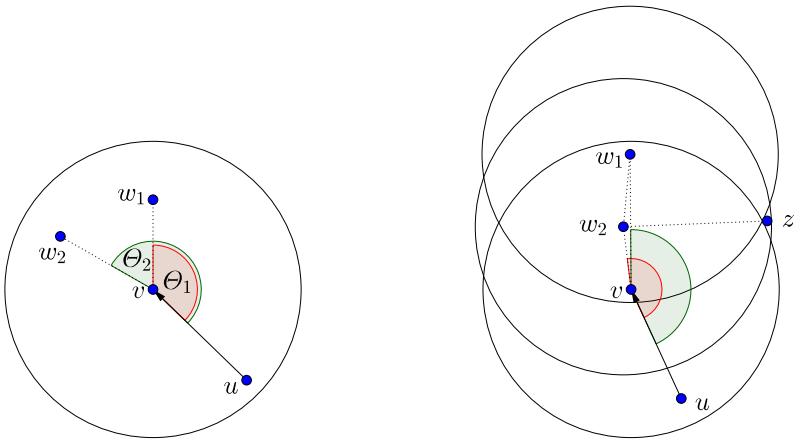


Figure 4.7: (a) Illustration of the sweep line-based beaconless recovery approach. The delay of a node w_i is proportional to angle $\Theta_i = \measuredangle uvw_i$, where v is the current forwarder and u is the previous-hop. (b) Scenario where selection of the angle minimizing node w_1 leads to routing loop $\langle v \to w_1 \to w_2 \to v \rangle$.

Let v be the node currently holding the packet and let u be the *previous hop*, i.e., the node which just forwarded the packet to v. Node v locally broadcasts a REQUEST (also RTS) including its own as well as u's position. Upon receiving of this REQUEST, all other nodes $w_i \in \mathrm{N}_1(v)$ compute a delay timer, using a commonly known delay function whose input is the set of positions $\{v, u, w_i\}$. The delay function maps the angle $\measuredangle uvw_i$ to a fraction of time slice $\mathrm{t}_{\max}$, a parameter which is either globally known to the nodes or part of the REQUEST. Once a node's delay timer expires, it answers with a RESPONSE (also CTS), including its position. This node then becomes a candidate for forwarding and forwarder v reacts to the RESPONSE by immediately sending another message, which cancels all other nodes' delay timers.

Sweep line-based recovery *Clockwise-Relaying* (CR) [161], *Bypass* [32, 199], *No-Beacon FACE* (NB-FACE) [211], *Angular Relaying* (AR) [18, 107], and *angle-based routing* (ABR) [28] are sweep line-based approaches to the beaconless recovery problem. The following explanation is illustrated by Figure 4.7a.

Given current forwarder v and previous-hop u, the delay of any neighbor $w_i \in \mathrm{N}_1(v)$ is proportional to the counter-clockwise angle $\measuredangle uvw_i$. Graphically speaking, the first node hit by rotating the sweep line segment $\overline{vu}$ in a counter-clockwise direction around v, is the first node that answers in the contention process. In Figure 4.7a $\Theta_1 < \Theta_2$ and hence, the delay timer of w_1 expires earlier than that of w_2, and w_1 becomes candidate for forwarding.

In CR [161], Bypass [32, 199], and ABR [28], the node $w_i \in \mathrm{N}_1(v)$, $w_i \neq u, v$, which minimizes the angle $\measuredangle uvw_i$ is directly selected for forwarding, i.e., forwarding node v

immediately forwards the DATA to w_i after receiving this node's CTS. However, as illustrated in Figure 4.7b, this may lead to a routing loop ($v \to w_1 \to w_2 \to v$), since edge vw_1 is intersected by another edge w_2z, which is the only edge leading towards the destination.

To avoid routing along intersected edges, NB-FACE [211] and AR [18, 107] make use of an additional *protest phase*. In NB-FACE [211] the forwarder waits for a time period τ after having received the CTS by candidate w_i before it announces a SELECTION of w_i. Within this time period nodes $w_j \in \mathrm{D}(v, w_i)$ send a PROTEST message. The rationale behind this is that the existence of $w_j \in \mathrm{D}(v, w_i)$ implies that uv_i is no edge of the Gabriel graph and hence, is possibly intersected by other edges of the network graph. The number τ is, however, not further specified in [211], which is problematic as pointed out by Rührup et al. [18, Section VI-A]. If τ is proportional to a constant fraction of the entire rotation angle, then this could cause multiple PROTEST messages, or lead to selecting an edge which is not free of intersections. Otherwise, if τ is proportional to the full rotation, then arbitrarily many nodes in $\mathrm{N}_1(v)$ may send their CTS messages, as they may not have been able to overhear the CTS sent by w_i. Hence, in order for this algorithm to produce a correct recovery path, the selection of a next-hop requires transmission of $\Theta(n)$ many messages in the worst-case, since either $\Theta(n)$ PROTEST or CTS messages are transmitted.

AR [18, 107] is similar to NB-FACE [211], but avoids the aforementioned problem. Forwarder v sends the RTS including its position as well as the position of previous-hop u. Nodes in $w_i \in \mathrm{N}_1(v)$ start their timer proportional to the counter-clockwise angle $\measuredangle uvw_i$. Nodes w_i with $u \in \mathrm{D}(v, w_i)$ answer with an INVALID-CTS, which does not cause other nodes to suppress sending their CTS. Although these nodes w_i are no suitable forwarders (vw_i is no edge of the Gabriel graph), their existence is crucial to determination of actual Gabriel graph neighbors of forwarder v. All other nodes respond with a CTS, including their position, upon timer expiration. On receiving the first CTS by w_j, forwarder v immediately sends a SELECT, including the position of the respective candidate w_j. This message suppresses all other nodes' timers and scheduled transmissions of CTS messages, and initiates the *protest phase*. In this phase all nodes $w_k \in \mathrm{D}(v, w_j)$ start a delay timer proportional to $\measuredangle uvw_k - \measuredangle uvw_j$ which determines the order of protests (if any). Upon time-out of such a protest timer, a node w_k sends a PROTEST, including its position. This node w_k then immediately becomes selected as the next candidate by the forwarder and another protest phase starts for node w_k, and so on. The first candidate node for which no PROTEST is received is guaranteed to be the first Gabriel graph neighbor of v in counter-clockwise direction w.r.t. ray $\overline{vu}$. However, it is easy to construct worst-case examples, where all nodes in $\mathrm{N}_1(v)$ either send a CTS or a PROTEST message during algorithm execution. Thus, the worst-case message complexity of AR [18, 107] is $\Theta(n)$, too.

Sweep curve-based recovery The problems regarding the $\Theta(n)$ worst-case message complexity are remedied by sweep curve-based beaconless recovery. Graphically speaking, instead of rotating a sweep line, a sweep curve is rotated around the forwarder. This

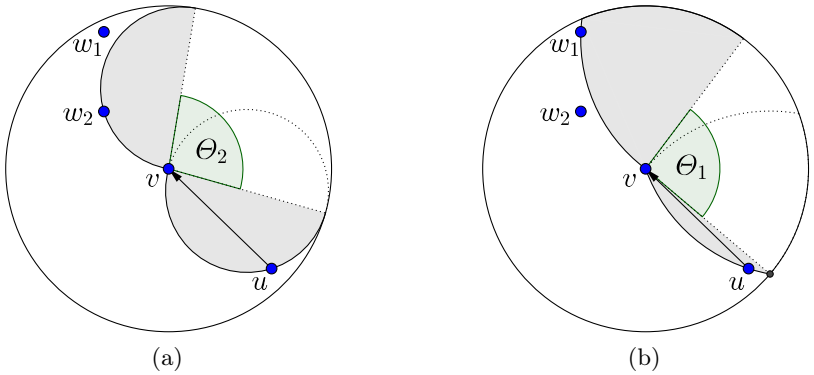


Figure 4.8: Illustration of the delay timer computation in the sweep curve-based recovery scheme Rotational Sweep (RS) using the (a) Sweep Circle (SC) and (b) Twisting Triangle (TT) delay functions. In both cases, the delay of a node w_i is proportional to the opening angle Θ_i which depends on the positions of the current forwarder v, previous-hop u, and candidate w_i. These figures are similar to [15, Figure 4 & 6].

idea was presented first by Kalosha and Rührup et al. in [18, 107] and later extended by Rührup et al. in a series of publications [15, 68, 69].

Rotational Sweep (RS) [15, 68, 69] is a beaconless recovery scheme that makes use of sweep curves. Generally, a forwarding node v locally broadcasts an RTS including its position, as well as the position of the previous-hop u. As in sweep-line recovery algorithms, neighbors $w_i \in \mathrm{N}_1(v)$ start a contention process based on a delay function, which can be represented by a sweep curve rotating in the counter-clockwise direction around the forwarder (see Figure 4.8). The node which is hit first by the sweep curve, i.e., the node whose timer expires first, answers with a CTS which includes its position. Upon reception of this CTS, forwarder v immediately broadcasts the DATA, which suppresses other nodes' timers and suppresses sending of other CTS messages.

RS proposes two delay functions. *Sweep Circle* (SC) is a half-circle attached to v, whereas *Twisting Triangle* (TT) is a Releaux triangle attached to v. The diameter of both SC and TT is equal to the unit disk radius. Initially, these geometric shapes are placed such that their boundary touches v and previous-hop u. Then, the counter-clockwise boundary curve of the respective shape is rotated in a counter-clockwise direction until the first node $w_i \in \mathrm{N}_1(v)$ is being hit. This can be expressed as a function of the positions of v, u, and w_i, such that the delay of a node w_i is proportional to the opening angle Θ_i (see Figure 4.8).

Let w_i denote the node which answered first with a CTS during RS. If SC is used for the contention process, then edge vw_i belongs to a planar supergraph of the Gabriel graph w.r.t. the underlying unit disk network graph. If instead TT is used for the contention process, then edge vw_i belongs to a supergraph of the relative neighborhood graph w.r.t. the underlying unit disk network graph. Although the edges used for recovery in the case

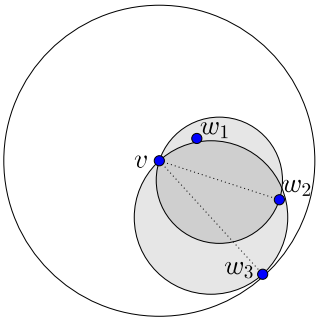


Figure 4.9: Illustration of the recovery routine of algorithm GDBF [187, 188]. After the RTS by executing node v, w_1 and w_3 respond with a CTS. Node w_2 is suppressed by the CTS of w_1.

of TT are not necessarily free of edge intersections, they are provably free of *critical edge intersections* that would lead to routing loops. Hence, both approaches yield guaranteed delivery recovery paths.

Let T_{SC} and T_{TT} denote the recovery paths (also called *traversals*) produced by RS on a particular local minimum situation when using SC and TT, respectively. The traversals T_{SC} and T_{TT} can be partially node disjoint, i.e., nodes used in T_{SC} may not be used in T_{TT} and vice versa. For an example see Figure 4.8, where for the exact same situation SC selects node w_2, whereas TT selects node w_1.

Most importantly, independent of the choice of the sweep curve and the actual input graph, RS requires only three messages for selecting the next-hop and forwarding the data packet. Hence, this algorithm's worst-case message complexity is $\mathcal{O}(1)$.

Topology-based recovery The non-aggressive contention Greedy mode of *Guaranteed Delivery Geographic Routing* (GDBF) [187, 188] has been discussed in Section 4.2. In the following, its two staged Recovery mode is presented. See Figure 4.9 for an illustration. Let v be a node in Recovery mode. In the first stage, v constructs a superset of its Gabriel graph neighborhood. To do so, node v locally broadcasts an RTS including its position. Neighbors $w_i \in \mathrm{N}_1(v)$ start a delay timer, where the delay function maps their Euclidean distance $\|w_i v\|$ proportionally to a fraction of time slice $\mathrm{t}_{\max}$. Upon timer expiration, a node locally broadcasts a CTS including its position. If a node w_j overhears a CTS from a node w_i with $w_i \in \mathrm{D}(w_j, v)$ during its delay (i.e., if w_j learns that it is not a Gabriel graph neighbor of v), then it cancels its delay timer and refrains from sending a CTS. This way, after time $\mathrm{t}_{\max}$ a superset of the Gabriel graph neighborhood of v has responded with a CTS.

For an example consider Figure 4.9, where the only Gabriel graph neighbor of v is node w_1. The CTS by w_1, whose delay timer expires first, suppresses the CTS from w_2. Node w_2 is said to be a *hidden node*, as it remains invisible for all other nodes. Then, although w_3 is not a Gabriel graph neighbor of v, it responds with a CTS, since w_2—the only witness for edge vw_3 not being included in the Gabriel graph—is hidden. During this stage the superset $\{w_1, w_3\}$ of u's Gabriel graph neighborhood responds with a CTS.

In the second stage, forwarder v selects a candidate next-hop w_i from the neighbors that are known from the first stage according to the left-hand rule. That is, it selects the node w_i that minimizes the counter-clockwise angle $\measuredangle uvw_i$. Then, it locally broadcasts this decision via a SELECT message. Nodes $w_j \in \mathrm{N}_1(v)$ with $w_j \in \mathrm{D}(v, w_i)$, if any, start a delay timer proportional to distance $\|vw_j\|$. The node with the shortest delay locally broadcasts a STOP message, which suppresses other nodes' timers and STOP message transmissions, and causes v to repeat the process with the next candidate in the counter-clockwise direction. If there is no node in $\mathrm{D}(v, w_i)$, then after time $\mathrm{t}_{\max}$ no STOP has been sent and w_i is the next-hop. Since only valid Gabriel graph neighbors are selected for forwarding, namely those according to the left hand rule, the resulting recovery path guarantees message delivery in combination with the Greedy routing mode. However, as in the case of the sweep line-based recovery algorithm AR [18, 107], it is easy to construct worst-case examples where the selection of the next Gabriel graph edge requires each node in $\mathrm{N}_1(v)$ to send at least one CTS or STOP message. Therefore, this algorithm's worst-case message complexity is $\Theta(n)$.

Non-planar forwarding and partial two-hop knowledge *Azimuth-Range ROuting for large-scale Wireless sensor networks* (ARROW) [212] differs from the previously described approaches in so far as it does not determine edges of specific planar subgraphs. Instead, the potential next-hop is determined, such that (1) it does not intersect one of the two previous edges used for forwarding, and (2) there is no hidden edge leading to a better forwarding node. The latter condition requires partial two-hop neighborhood information of the forwarding node.

Initially, the local minimum node v determines the forwarding direction (clockwise or counter-clockwise) using a simple contention process. Nodes $w_i \in \mathrm{N}_1(v)$ are delayed proportional to the angle $\measuredangle w_ivd$, or $\pi - \measuredangle w_ivd$ if $\measuredangle w_ivd$, where d is the packet's destination. The packet is forwarded to the node that minimizes this angle. The resulting forwarding direction is stored in the packet header and maintained henceforth. In addition, the packet always stores the positions of the two previous routing hops.

Whenever a node v in Recovery mode (i.e., after the first forwarding step by the local minimum node) has received the packet from the previous-hop u, the next-hop is determined as follows (see Figure 4.10 for an illustration). At first the current forwarder starts a contention process by broadcasting an RTS. Nodes $w_i \in \mathrm{N}_1(v)$ are delayed proportional to angle $\measuredangle w_ivu$ w.r.t. the direction stored in the packet header. Only particular nodes w_i participate. Note that given the positions of the previous hop u and second-last hop u', any node w_i can easily determine if edge vw_i intersects uu' or uv. Only those nodes w_i which do not intersect these edges (i.e., satisfy the so called *backward rule*) participate in the contention process. The node satisfying the backward-rule and minimizing the angle is the first to reply with a CTS and is forwarded the data packet. However, this simple angle-based choice may lead to routing loops (recall the situation depicted in Figure 4.7b) and hence, it must be checked if edge vw_i is intersected by a hidden edge, which leads to a better forwarding node (see edge w_2y in Figure 4.10). To do so, the node currently holding the packet applies the *interconnected triangle rule*.

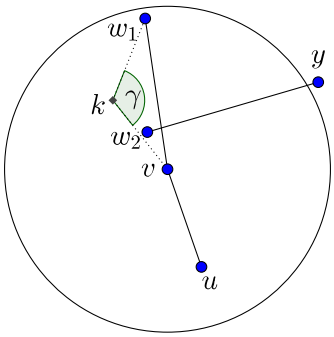


Figure 4.10: Illustration of *IC triangle rule* during algorithm ARROW [212]. For the next recovery edge vw_1 it is checked if the isosceles triangle with $\gamma = 2\pi/3$ contains a node w_2 which is connected to a node y, s.t. w_2y intersects the forwarding edge vw_1.

Essentially, for any node such as w_2 which is located in the isosceles triangle $\triangle(v, w_1, k)$, with $\measuredangle vw_1k = \measuredangle w_1vk = \pi/6$ and $\measuredangle vkw_1 = 2\pi/3$, it is checked if w_2 is connected by an edge to a node y s.t. w_2y intersects edge vw_1. In case such an edge exists, it is used for recovery (in case there are multiple nodes w_j in triangle $\triangle(v, w_1, k)$ and/or multiple intersecting neighbors y_i, then the combination of nodes maximizing $\measuredangle vw_jy_i$ is used for forwarding) and the interconnection triangle rule is applied again. Otherwise, edge vw_1 was a correct choice and the forwarding process is restarted at node w_1. The recovery path resulting from ARROW [212] is loop-free, provides positive progress towards the destination, and therefore, provides delivery guarantee in combination with Greedy routing. In contrast to the previous outlined beaconless recovery algorithms which guarantee delivery, in ARROW the nodes do not need to know their communication radius for computation of their delay timers.

In the worst-case, there are $\mathcal{O}(n)$ nodes located in the isosceles triangle, and each one has to transmit at least one message to determine whether it has an adjacent edge that intersects vw_1. It is easy to construct scenarios, where all such nodes actually transmit at least one message. ARROW thus has a worst-case message complexity of $\Theta(n)$.

In PF/PFMAC [172, 173]—described previously in the context of aggressive contention Greedy routing algorithms in Section 4.2—a forwarding node v encounters a local minimum situation if the higher priority sectors (p_3 and p_2 in Figure 4.3) are empty of nodes. The Greedy contention function ensures that nodes in the recovery sectors, which have priority p_1, contend at least as long as nodes in higher priority sectors. In addition, this delay function favors those nodes $w_i \in \mathrm{N}_1(v)$ in the recovery sectors which maximize the Euclidean distance to forwarder v, while minimizing the angle Θ_i, the inner angle formed by the line segment $\overline{w_iv}$ and the orthogonal straight line to vd passing through v (see Figure 4.3). Upon timer expiration, nodes in the recovery sectors transmit a so called SOLICITATION message, which suppresses other such messages in their sectors. That is, at most two such messages are being transmitted. These messages initiate a contention process in the neighborhoods of their senders using the same delay functions as in Greedy

mode. Only one SOLICITATION-RESPONSE is allowed per sector. After that process, senders of the SOLICITATION messages know at most six candidates and select the most suitable one (e.g., based on progress towards the destination) and send the corresponding node position to the current forwarder v. Upon receiving this information, it selects the most suitable two-hop neighbor and forwards the message accordingly. Since the algorithm makes use of heuristics only, it cannot guarantee progress along a recovery path and hence, it does not provide delivery guarantee. However, the selection of the next two forwarding hops requires at most a constant number of message transmissions and therefore, this algorithm's worst-case message complexity is $\mathcal{O}(1)$.

The author of this thesis is well aware of *Adaptive Load-Balanced Algorithm, Rainbow version* (ALBA-R) [30] that may be used for beaconless recovery. However, it is not stateless and therefore of no further relevance for this work.

Taxonomy Table 4.3 lists and summarizes the aforementioned beaconless recovery routing algorithms in chronological order. For each algorithm, this table lists the following:

k-locality refers to the degree of k-locality required by nodes to compute the next-hop.
UDG is checked with '✓' if the algorithm is designed for operation in unit disk graphs. Otherwise, it is marked with a '✗'.
Guaranteed delivery is checked with '✓' if the resulting recovery path always satisfies the progress criterion. Otherwise, it is marked with a '✗'.
Message complexity refers to the algorithms worst-case message complexity.
Reference refers to source(s).
Year lists the earliest point in time of publication.

Algorithm	k-locality	UDG	Guaranteed delivery	Message complexity	Reference	Year
CR	1	✓	✗	$\mathcal{O}(1)$	[161]	2004
Bypass	1	✓	✗	$\mathcal{O}(1)$	[31, 32]	2005
NB-FACE	1	✓	✓ [a]	$\Theta(n)$ [a]	[211]	2006
GDBF	1	✓	✓	$\Theta(n)$	[187, 188]	2006
AR	1	✓	✓	$\Theta(n)$	[18, 107]	2008
PF/PFMAC	2	✗	✗	$\mathcal{O}(1)$	[172, 173]	2010
RS-SC,RS-TT	1	✓	✓	$\mathcal{O}(1)$	[15, 68, 69]	2010
ABR	1	✗	✗	$\mathcal{O}(1)$	[28]	2010
ARROW	2	✓	✓	$\Theta(n)$	[212]	2013

[a] Guarantees message delivery only if timers reflect full sweep rotations, in which case the message complexity is $\Theta(n)$.

Table 4.3: Taxonomy of beaconless recovery algorithms in chronological order.

It is important to note that the delivery guarantees of the above protocols is only provided if the input graph is a unit disk graph. Except for ARROW [212], nodes need to know the unit transmission range R.

4.4 Multicast Routing

Geographic multicast routing refers to the task of routing a packet from a single source to a set of network nodes, the *multicast group*. This set of destination nodes as well as their positions is assumed to be known in advance and explicitly addressed in the packet header. A multicast algorithm *succeeds* or is said to *guarantee delivery*, if all addressed nodes actually receive the packet. For a general overview on geographic multicast algorithms, the reader is referred to the survey from Boukerche et al. [91].

Beacon-less geographic routing for multicast applications (BRUMA) [22] is the first beaconless algorithm for geographic multicast and the only one that guarantees message delivery without making use of beacon-based recovery. Let v be a node that wants to send a message to a set of multicast destinations $D = \{d_0, ..., d_k\}$ specified by their geographic positions. Node v first computes the *Euclidean Minimum Spanning Tree* (see Definition 2.10) $\Delta_v = \text{EMST}(D \cup \{v\})$ and computes the set $R(v)$ of *first level receivers*, i.e., the subset of destinations from D to which v is connected by an edge in Δ_v. In a second step, for each first level receiver $r_i \in R(v)$, node v uses a BOSS-like contention process (see non-aggressive Greedy routing algorithms in Section 4.2) in order to forward the packet to one neighbor $w_i \in \text{N}_1(v)$, which has correctly received the packet while maximizing the progress w.r.t. r_i. If no such neighbor exists, i.e., if v is a local minimum node w.r.t. r_i, then the beaconless recovery strategy from GDBF [187, 188] (see Section 4.3) is applied in order to route the message to the corresponding first level receiver. The costs for forwarding a packet per hop are constant in Greedy mode, whereas in Recovery mode they are $\Theta(n)$ in the worst-case. This is due to the same worst-case message complexity of the recovery routine of algorithm GDBF [187, 188]. In addition, subroutine GDBF assumes unit disk input graphs.

Receiver-based multicast (RBMulticast) [24] and the *distributed multicast protocol based on beaconless routing* (DMPB) [213] follow the same idea. A forwarding node v partitions the plane into four quadrants with the origin at v. For each quadrant containing at least one multicast destination, the packet is duplicated and assigned the corresponding multicast destinations. The destinations of these packets are virtual nodes at the barycenters of the multicast destinations in the respective quadrant. A beaconless Greedy routing strategy is used to forward the packets towards the virtual nodes. In RBMulticast, the node with largest progress towards the virtual node is selected for forwarding. In DMPB, forwarder v partitions the plane s.t. the barycenter of the multicast destinations lies on the angle bisector of one of the four quadrants. Local minimum situations are not considered by DMPB. Hence, DMPB does not provide guaranteed delivery and only requires a constant number of message transmissions per hop, depending on the choice of the actual beaconless Greedy algorithm. In RBMulticast it is assumed that local minimum situations can be resolved using the beacon-based recovery algorithm GPSR [37]. Hence, provided that the network graph obeys the unit disk graph model, RBMulticast actually provides guaranteed delivery, but at the cost of producing $\Theta(n)$ many message transmissions per recovery hop in the worst-case.

Geographic multicast (GEM) [214] is a two-phased Greedy scheme for geographic multicast. In the first phase, at most two directions are determined by the forwarding

node v that point at positions of optimal next hops. To do so, it computes the average distance between its position and the multicast destinations $D = \{d_0, ..., d_k\}$. The difference between this value and the corresponding value for some other point in v's unit disk gives a notion of progress w.r.t. the destinations. Now depending on set D, the algorithm determines either one or two points on the unit disk boundary of v that maximize the progress. In order to keep the computation feasible, only points with fixed angular distances are taken into consideration. In the second phase, forwarder v determines the next hop for each optimal direction using a Greedy routing contention process. The progress criterion is the nodes' projected progress on the optimal directions. The greater the progress, the shorter are the nodes' delay timers. GEM assumes dense networks in which no local minimum situations occur and recovery is not required. Therefore, this algorithm cannot provide a delivery guarantee for arbitrary unit disk input graphs. The costs for forwarding the packet are bounded by a constant.

Taxonomy Table 4.4 lists and summarizes the aforementioned beaconless multicast algorithms in chronological order. For each algorithm, this table lists the following:

UDG is checked with '✓' if the algorithm is designed for operation in unit disk graphs. Otherwise, it is marked with a '✗'.
Guaranteed delivery is checked with '✓' if the algorithm provides delivery guarantee. Otherwise, it is marked with a '✗'.
Recovery strategy specifies which strategy is used for escaping local minimum situations, if any.
Message complexity refers to the algorithms worst-case message complexity.
Reference refers to source(s).
Year lists the earliest point in time of publication.

Algorithm	UDG	Guaranteed delivery	Recovery strategy	Message complexity	Ref.	Year
BRUMA	✓	✓	GDBF [187, 188]	$\Theta(n)$	[22]	2009
RBMulticast	✓	✓	GPSR [37]	$\Theta(n)$	[24]	2012
DMPB	✗	✗	-	$\mathcal{O}(1)$	[213]	2013
GEM	✓	✗	-	$\mathcal{O}(1)$	[214]	2013

Table 4.4: Taxonomy of beaconless multicast algorithms in chronological order.

4.5 Topology Control

The task of a *beaconless topology control algorithm* is to provide its executing node with its local view on the desired topology, without prior determination of this node's network neighborhood, i.e., without use of beaconing. To be more specific, given a network graph $G = (V, E)$ and a desired topology $G' = (V', E')$ of G, the task of a beaconless topology

control algorithm is to provide its executing node $v \in V$ with its adjacency $E_{G'}(v)$ without making use of beaconing on the underlying network graph G.

Note that the above definition is only a special case of the more general definition of *reactive local topology control* introduced in Chapter 5

Until now, only few beaconless topology control algorithms have been proposed. These focus on local view constructions on connected and planar subgraphs of unit disk graphs $\mathrm{UDG}(V)$. Before these algorithms are presented in detail, the generic scheme underlying these is provided.

***Generic beaconless topology control scheme* executed by a node v:**

1. At algorithm start, v is unaware of its network neighborhood $\mathrm{N}_1(v)$. It starts a *selection phase* by locally broadcasting a REQUEST (or RTS) including its geographic position. This REQUEST is overheard by all nodes in $\mathrm{N}_1(v)$. Node v schedules termination of this selection phase after a delay of $\mathrm{t}_{\max}$, a fixed time slice that is either globally known to all nodes, or specified in the REQUEST.

2. Upon reception of a REQUEST sent by v, any neighbor $u \in \mathrm{N}_1(v)$ schedules sending of a RESPONSE (or CTS), which includes its geographic position, after a certain delay. Typically this delay is $(\|uv\|/\mathrm{R}) \cdot \mathrm{t}_{\max}$, where R denotes the unit disk radius.

3. Upon expiry of the delay timer, a node $u \in \mathrm{N}_1(v)$ locally broadcasts its RESPONSE which is overheard by all nodes in $\mathrm{N}_1(u)$, in particular by v. If a node $u \in \mathrm{N}_1(v)$ overhears during its delay a RESPONSE of some node w with $w \in \mathrm{PR}(v,u)$, where $\mathrm{PR}(v,u)$ denotes the *proximity region* (which depends on the particular topology; see Figure 2.4 for details), then node u either extends its delay to some larger fraction of $\mathrm{t}_{\max}$, or cancels the scheduled transmission of its RESPONSE.

4. After time of at most $\mathrm{t}_{\max}$, only a subset of candidates $\mathcal{N} \subseteq \mathrm{N}_1(v)$ has actually transmitted a RESPONSE. Nodes in $\mathrm{N}_1(v) \setminus \mathcal{N}$, i.e., nodes that have not sent any message, are called *hidden nodes*, for they remain invisible. Depending on the delay function and the proximity region, either $\mathcal{N}$ is the desired local view $E_{G'}(v)$ of node v in G', or not. In the former case the topology control task is completed. In the latter case, an additional *protest phase* is needed which removes those candidates from $\mathcal{N}$ that do not belong to v's local view on the desired topology. Typically, this protest phase is similar to the selection phase in the sense that few nodes decide to transmit a PROTEST message during a contention process.

Next, the individual algorithms are described in detail.

Recall the routine for handling local minimum situations of *Guaranteed Delivery Geographic Routing* (GDBF) [187, 188], which has been described in detail in Section 4.3. With the following simple modifications it can easily be converted into a reactive topology control algorithm for computation of a node's adjacency in the Gabriel graph of the underlying unit disk graph.

A node v starts the execution of the recovery routine of GDBF and obtains a superset $\mathcal{N}$ of its Gabriel graph neighborhood.[5] Nodes in $\mathcal{N}$ are referred to as *candidates*. Nodes $c_i \in \mathcal{N}$ are sorted, e.g., according to their counter-clockwise angle $\measuredangle uvc_i$, where ties are broken such that the node which is further away is processed earlier. For each edge vc_i in this order, the PROTEST message-based test of whether edge vc_i belongs to the Gabriel graph neighborhood of v is performed. Each PROTEST message is used by v to sort out candidates from $\mathcal{N}$ which do not belong to its Gabriel graph neighborhood. That is, each PROTEST message removes one or several candidates from $\mathcal{N}$. The resulting set of candidates is the set of Gabriel graph neighbors of v w.r.t. the underlying unit disk network graph. The message complexity of this approach will be discussed along with the one from BFP, which is presented next.

Beaconless Forwarder Planarization (BFP) [18, 107] generalizes and improves the aforementioned approach. Essentially, instead of performing one protest phase per candidate, it only performs one protest phase for all candidates. Besides the Gabriel graph (GG), the algorithm can alternatively be used to construct a node's local view on the *Circlunar Neighborhood Graph* (CNG), or *Relative Neighborhood Graph* (RNG) by using the corresponding proximity areas (see Figure 2.4). In the remainder of this thesis, the instances of BFP which construct a node's local view on GG, RNG, and CNG are denoted by BFP-GG, BFP-RNG, and BFP-CNG, respectively.

The *selection phase* of BFP is similar to that of GDBF. Initiating node v locally broadcasts an RTS, including its position, and starts a delay timer on input $\mathrm{t_{max}}$. Neighbors $u \in \mathrm{N_1}(v)$ schedule sending of a CTS, including their position, after a delay of $(\|uv\|/\mathrm{R}) \cdot \mathrm{t_{max}}$. If a node $u \in \mathrm{N_1}(v)$ overhears during its delay a CTS from a node w with $w \in \mathrm{PR}(v, u)$ (i.e., w is located in the proximity region w.r.t. v and u), then u becomes a hidden node, cancels its delay timer as well as its CTS transmission, and continues listening to message transmissions. In case a hidden node u overhears a CTS by a node w with $u \in \mathrm{PR}(v, w)$, then u adds w to its list of *violating nodes*.

After the selection phase, i.e., after time $\mathrm{t_{max}}$, the *protest phase* begins. Nodes with non-empty lists of violating nodes schedule sending of a PROTEST message, including theirs as well as the positions of the violating nodes, using the same delay function as in the selection phase. A node protests against a node w only if it has not overheard a protest against w in the meantime.

After time at most $\mathrm{t_{max}}$, all PROTEST messages have been sent. Nodes that have sent a CTS and for which no PROTEST has been received by the initiating node v constitute its neighborhood in the corresponding proximity graph.

One major drawback of distance-based delay functions in general is the collision of messages of equidistant nodes. If there are two or more neighbors that are equidistant to the executing node v, then their timers expire at the same moment and their messages collide. In fact, this is a non-trivial problem, since the order of message arrivals is crucial w.r.t. the algorithm's correctness. Therefore, Rührup et al. [18, 107] assume that the pairwise node distances in the underlying unit disk graph are distinct. This assumption,

[5] The positions of destination node d and previous-hop u can be chosen arbitrarily without invalidating the correctness of this algorithm.

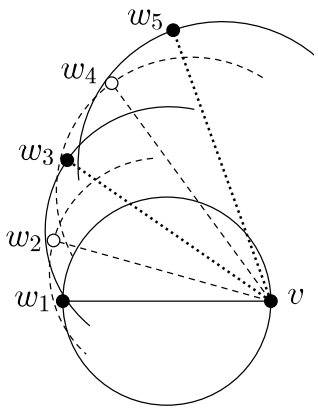


Figure 4.11: Example where GDBF and BFP-GG produce $\Theta(n)$ message transmission during execution by node v. Hidden nodes are white and are connected by dashed edges. Edges of non-hidden nodes violating the GG-rule are dotted. The solid line represents the only GG-edge. This figure is similar to [18, Figure 4].

however, has notable implications. Whereas GG still has a maximum node degree of $\mathcal{O}(n)$, the maximum node degree of RNG and CNG is bounded by 5 and 13, respectively, under this assumption [18]. This observation has to be kept in mind while considering BFP's message complexity, which is discussed next.

The construction of a node's Gabriel graph neighborhood using algorithm GDBF [187, 188] or BFP-GG [18, 107] requires $\Theta(n)$ message transmissions in the worst-case. To see this, consider the unit disk graph $G = (V, E)$ depicted in Figure 4.11 which has previously been defined by Rührup et al. for proving [18, Theorem 2]. It is defined as follows:

- $V = \{v, w_1, ..., w_{n-1}\}$,
- $\mathrm{D}(v, w_1) \cap V \setminus \{v, w_1\} = \emptyset$,
- $\forall\, 1 < i \leq n-1 : \mathrm{D}(v, w_i) \cap V \setminus \{v, w_i\} = w_{i-1}$, and
- $\forall\, u, v \in V,\ uv \in E$.

Consider the execution by node v. At first, node w_1 transmits a CTS as it is closest to v. Its CTS suppresses w_2, since $w_1 \in \mathrm{D}(v, w_2)$. Since w_2 is a hidden node, eventually w_3 transmits its CTS, which suppresses w_4, and so on. In general, nodes having an odd index i transmit a CTS and thereby suppress nodes having an even index $i + 1$.

During the protest phase, each node with index i being even has to transmit exactly one PROTEST message, since it is the only node witnessing the violation of the Gabriel graph proximity rule w.r.t. edge vw_{i+1}. Hence, in total each node $w_j \in V$, $1 \leq j \leq n-1$, either transmits a CTS or a PROTEST message. Since there may be arbitrarily many nodes w_i, construction of the single Gabriel graph edge vw_1 requires $\Theta(n)$ message transmissions in the worst-case.

The number of message transmissions required for construction of a node's neighborhood in RNG and CNG using BFP-RNG and BFP-CNG, respectively, is substantially different. In Corollaries 2 & 3 in [18], Rührup et al. prove that during execution of BFP-RNG and BFP-CNG by any node v, at most a constant number of CTS and PROTEST messages are being transmitted, and there exists a scenario where this upper bound is actually reached. Therefore, the number of message transmissions is $\Theta(|\mathrm{N}_{\mathrm{RNG}}(v)|)$ and $\Theta(|\mathrm{N}_{\mathrm{CNG}}(v)|)$, respectively, where $|\mathrm{N}_H(v)|$ denotes the number of neighbors (node degree) of v in graph H. Since the maximum node degree of a node in RNG and CNG is bounded by a constant under the assumption that any two node distances are distinct in the input graph, in fact both BFP-RNG and BFP-CNG have message complexity $\mathcal{O}(1)$.

Taxonomy Table 4.5 lists and summarizes the aforementioned beaconless topology control algorithms in chronological order. For each algorithm, this table lists the following:

Topology control structure specifies the topology for which the local view of a node is constructed.
Message complexity refers to the worst-case number of message transmissions during algorithm execution.
Node degree refers to the maximum degree of a node in the constructed topology.
Reference refers to source(s).
Year lists the earliest point in time of publication.

Algorithm	Topology control structure	Message complexity	Node degree	Reference	Year
GDBF [a]	GG	$\Theta(n)$	$\mathcal{O}(n)$	[187, 188]	2006
BFP-GG	GG	$\Theta(n)$	$\mathcal{O}(n)$	[18, 107]	2008
BFP-CNG	CNG	$\Theta(\|\mathrm{N}_{\mathrm{CNG}}(v)\|)$	13	[18, 107]	2008
BFP-RNG	RNG	$\Theta(\|\mathrm{N}_{\mathrm{RNG}}(v)\|)$	5	[18, 107]	2008

[a] GDBF refers here to the modified variant of the recovery strategy as described in Section 4.5

Table 4.5: Taxonomy of beaconless topology control algorithms in chronological order.

Chapter 5

Foundation of Reactive Local Topology Control

In this chapter the concept of *reactive local topology control*[1] is introduced and shown to be meaningful. It is a generalization of what has previously been called *beaconless topology control* (see Section 4.5), a term which is both too narrow to capture all possible configurations of topology control algorithms that forgo beaconing, and too broad to properly distinguish the elements which it subsumes. In addition, this notion is simply too vague for in-depth investigations which topology control problems are principally solvable if use of beaconing is disallowed.

In Section 2.4, *topology control* has been introduced as the art of transforming a given wireless network graph $G = (V, E)$ into another representation $G' = (V', E')$, which obeys certain desired properties such as connectivity, planarity, and low stretch.

Beaconless topology control has been described in Section 4.5 as the task of computing the local view (adjacency) of a node $v \in V$ in the desired topology G' without making use of beaconing.

In all existing beaconless topology control algorithms (see Table 4.5), the desired topology G' is always a subgraph of network graph G, such that $G' = (V, E' \subseteq E)$. However, there are local algorithms (e.g., [51, 70, 71, 136, 137]) which can be used to obtain a node's local view on planar topologies that are not subgraphs of the underlying network graph. For instance, the planar backbone graph $Virt(G_b)$ introduced by Lillis et al. [70, 71] (see Section 3.2) is composed of *physical nodes* that actually belong to the underlying QUDG, as well as of *virtual nodes* which do not belong to it. If $Virt(G_b)$ is used, e.g., for recovery in a local minimum situation occurring during local geographic routing, it may be necessary to obtain the adjacency of a backbone node v in $Virt(G_b)$ on demand. But what if this node v is not a node from the set of physical nodes, but a virtual node? Then a physical *proxy* $u \in V$ is required that computes v's local view on $Virt(G_b)$ on behalf of v. As shown later in this thesis (see Chapter 8), local views on such graphs can also be constructed in a beaconless fashion. However, such approaches are not covered by the notion of *beaconless topology control* as considered so far, but they are covered by the concept of *reactive local topology control* introduced here.

[1] The term *local* in *reactive local topology control* emphasizes the focus on *local algorithms* as given in Definition 2.5, whereas the term *reactive* is used in the meaning ascribed to it by Frey and Rührup in [14]: they label a (topology control) algorithm as *reactive*, if it can be executed by any node on demand, without any neighborhood knowledge, and without making use of beaconing. Henceforth, the term *reactive* is used with this meaning.

Furthermore, the set of existing beaconless topology control algorithms is heterogeneous w.r.t. message complexity in the following sense. As presented in Section 4.5, for construction of the local view of some node v, algorithms BFP-RNG and BFP-CNG require $\Theta(|\mathrm{N}_{\mathrm{RNG}}(v)|)$ and $\Theta(|\mathrm{N}_{\mathrm{CNG}}(v)|)$ message transmissions, respectively, where $|\mathrm{N}_H(v)|$ denotes the number of neighbors (node degree) of v in graph H. Informally speaking, the ratio of the number of message transmissions to the number of constructed neighbors in the desired topology is always bounded by a constant. In contrast, algorithms BFP-GG and GDBF require $\Theta(n)$ message transmissions in the worst-case; therefore, the aforementioned ratio is not bounded by a constant, but depends on the number of network nodes.

This observation suggests that there are different classes of beaconless topology control algorithms, at least from a message complexity point of view.

The concept *reactive local topology control* remedies all of the aforementioned shortcomings as follows. It generalizes and formalizes the problem statement underlying beaconless topology control. Based thereon, the general classes of $\mathcal{O}$*-reactive* and Ω*-reactive* topology control algorithms are defined by means of message complexity. These unambiguously distinguish reactive from conventional, beacon-based, distributed algorithms, and simultaneously serve as taxonomy for all existing and prospective algorithms of this kind. Finally, the concept actually facilitates in-depth investigations of the power of the beaconless approach. This is demonstrated by proving two fundamental propositions: the impossibility of $\mathcal{O}$-reactive computation of local views on Gabriel graphs and grid graphs.

5.1 Preliminaries

Unless stated differently, in this chapter wireless networks are modeled as *general graphs* $G = (V, E)$. V is a set of n network nodes and E is a set of undirected *edges* representing bidirectional communication links. The particular algorithms considered in this chapter require that $V \subset \mathbb{R}^2$, this is, however, of no further relevance for the general concept of reactive local topology control.

In order to keep this concept as general as possible and not to limit it to problems related to geometric graphs, in this chapter a classical *addressing model* is assumed, where nodes are equipped with globally unique node identifiers (IDs). Each such ID (or node address) requires $\Omega(\log n)$ bits for its representation. Unless stated differently, it is assumed that a message sent by a node always contains this node's ID and possibly its geographic position, if applicable. Hence, any message is of size $\Omega(\log n)$ bits. Furthermore, the size of a message (e.g., in terms of number of bits) is generally not restricted. Nevertheless, the algorithm classes introduced in Section 5.2 implicitly restrict the message size in terms of number of bits, by a function of the number of nodes n.

Local unicast [*local broadcast*] refers to sending of a single message from a node $v \in V$ to one [all] of its neighbors $w \in \mathrm{N}_1(v)$. It is assumed that transmissions are reliable, delivered instantaneously, and can be performed by any node at any given time.

A collection of graphs satisfying a certain common property is termed a *graph class*. One example of a graph class is $\mathcal{UDG}$, the class of *unit disk graphs*.

The *k-hop neighborhood* $\mathrm{N}_k(v)$ of a node $v \in V$ refers to v and all nodes $w \in V$, which are reachable from v along a path consisting of at most k edges. The *k-hop local view* $G[v,k]$ of v on graph $G = (V,E)$ is a subgraph of G whose node set is $\mathrm{N}_k(v)$ and whose edge set is given by $\{xy \in E : x \in N_{k-1}(v) \text{ and } y \in N_k(v)\}$. Note that edges to nodes that are more than k hops apart are not included.

The number of nodes in $G[v,k]$ is denoted by $|G[v,k]|$. The 1-hop local view $G[v,1]$ is abbreviated by $G[v]$, whenever this is clear from the context.

Finally, in this chapter *message complexity* of a distributed algorithm is defined as the total volume of communication measured in bits, i.e, it refers to the total number of bits transmitted by all nodes during algorithm execution over every legal input in the worst-case. Note that this is slightly different from Definition 2.6, where message complexity simply refers to the number of messages transmitted during algorithm execution.

5.2 Formalization of Reactive Local Topology Control

5.2.1 Local Topology Mappings and Local View Topology Control

Formally, a topology control algorithm is a mapping function that maps a given topology into another one.

Definition 5.1 (Topology mapping)**.** A *topology mapping* is a function $\tau : \mathcal{C} \to \mathcal{D}$, where $\mathcal{C}$ and $\mathcal{D}$ are graph classes. For a graph $G \in \mathcal{C}$, graph $\tau(G) \in \mathcal{D}$ is called the *image* of G, whereas G itself is the *pre-image* of image $\tau(G)$.

This work concentrates on *local topology mappings*. Informally speaking, a topology mapping is k-local for some constant $k \geq 1$, if the local view of any node v from the image graph $\tau(G)$ can be constructed by a k-local algorithm, i.e., based solely by considering this node's k-hop neighborhood information. For instance, Gabriel graph (GG) and partial Delaunay triangulation (PDT) are 1-local topology mappings, whereas the 2-localized Delaunay triangulation ($\mathrm{LDel}^{(2)}$) is a 3-local topology mapping. These topology mappings are functions τ, such that the following two operations yield indistinguishable results:

Operation 1 Apply τ on the pre-image graph G and then obtain v's local view $\tau(G)[v]$.

Operation 2 Restrict the pre-image graph G to the k-neighborhood of v, i.e., consider $G[v,k]$, then apply τ on $G[v,k]$, and finally obtain v' local view $\tau(G[v,k])[v]$.

In the virtual nodes example given in the beginning of this chapter, node $v \in \tau(G)$ is not necessarily a physical node and consequently not necessarily a computational entity. Therefore, in order to keep the definition as general as possible, the following definition allows that the local view of a node v from the image graph may be computed based on k-hop neighborhood information by an actual physical node u from the pre-image graph. For the concepts introduced in this chapter it is of no further relevance how nodes u and v are actually related; typically, they are identical.

Definition 5.2 (k-local topology mapping)**.** A topology mapping $\tau : \mathcal{C} \to \mathcal{D}$ is termed *k-local topology mapping*, if for each $G \in \mathcal{C}$ and every node $v \in \tau(G)$ there exists a node $u \in G$ such that $\tau(G)[v] = \tau(G[u, k])[v]$.

The general problem of determining a node's local view in a desired topology can be defined as follows.

Definition 5.3 (Local view topology control)**.** *Local view topology control* refers to the task of computing a local view on a given topology mapping $\tau : \mathcal{C} \to \mathcal{D}$ from a class of communication graphs $\mathcal{C}$ to a class of image graphs $\mathcal{D}$. That means, given a communication graph $G \in \mathcal{C}$ and a node v in $\tau(G)$, a specific node $u \in G$ has to determine the local view $\tau(G)[v]$.

Reactive local topology control can be considered a special case of *local view topology control*. Based on the observation that existing algorithms differ w.r.t. message complexity, two classes can be distinguished that are introduced next.

5.2.2 $\mathcal{O}$-Reactive Topology Control

Algorithms like BFP-RNG and BFP-CNG [18, 107] have in common that the number of message transmissions that are required for computation of a node's local view, is in the order of the number of incident edges in this local view; one initial local broadcast, one response for each edge in the local view, and possibly a constant additive overhead related to a protest phase.[2]

The following definition combines the principal idea and scheme underlying beaconless algorithms like BFP-RNG and BFP-CNG, generalized to k-hop neighborhoods on the one hand, and the lower bound message size of distributed algorithms on the other hand.

Definition 5.4 ($\mathcal{O}$-reactive topology control)**.** A local view topology control algorithm for a k-local topology mapping $\tau : \mathcal{C} \to \mathcal{D}$ is called *$\mathcal{O}$-reactive* if the following applies. Let $G \in \mathcal{C}$ be any graph with n nodes. For any node $v \in \tau(G)$ there exists a node $u \in G$ such that the local view $\tau(G)[v]$ can be constructed, while the following holds:

- At algorithm start, nodes in G are unaware of their network neighborhoods (information from previous algorithm executions is assumed to be outdated and discarded).
- Node u starts the algorithm by a single local broadcast.
- The number of bits transmitted during algorithm execution is in the order of

  $$\mathcal{O}\left(|\tau(G)[v, k]| \cdot \log n\right) .$$

  Recall that $|\tau(G)[v, k]|$ denotes the number of nodes in $\tau(G)[v, k]$ and note that $|\tau(G)[v, k]| > 0$, because $v \in \tau(G)[v, k]$.

[2] These particular algorithms are designed for operation on geometric graphs, where nodes have distinct positions in the Euclidean plane and are aware of these geographic positions. Under these assumptions, geographic positions serve as unique IDs and therefore, no dedicated unique node IDs are assumed or required. Furthermore, the messages exchanged in these algorithms contain at most a constant number of geographic node positions.

Note that the above definition does not restrict the maximum message size; e.g., algorithms requiring only $\mathcal{O}(1)$ messages, each of size $\mathcal{O}(|\tau(G)[v,k]| \cdot \log n)$ bits, also match the above definition. Ultimately, reactive topology control algorithms are of interest since they reduce the overall communication load on the network. Thus, the total amount of exchanged bits of an algorithm execution is the actual bound of interest, not the size of an individual transmission.

5.2.3 Ω-Reactive Topology Control

The definition of $\mathcal{O}$-reactive topology control excludes topology control algorithms like BFP-GG [18, 107] and GDBF [187, 188], although they are reactive. This is due to worst-case scenarios previously discussed in Section 4.5. In these scenarios the total number of bits transmitted during algorithm execution is not bounded from above by $\mathcal{O}(|\tau(G)[v]| \cdot \log n)$, but is in the order of $\Omega(n \log n)$, where the $\log n$-term represents the assumption that each message is of size $\Omega(\log n)$ bits.

In general, however, these algorithms do better than a conventional, beacon-based approach, where all 1-hop neighbor addresses/positions are determined. If nodes are initially unaware of their network neighbors, determination of set $\mathrm{N}_1(u)$ by any node u presupposes that each node in $\mathrm{N}_1(u)$ transmits at least one message in order to introduce itself, which yields $|G[u]|$ messages in total. Each such message contains at least one node's address/position. Therefore, the lower bound message complexity in terms of number of bits is $\Omega(|G[u]| \cdot \log n)$.

This observation can be generalized to the k-hop case, i.e., approaches where node u has to determine its complete k-hop neighborhood before it can compute a local view on the topology. In this case, each node in $\mathrm{N}_k(u)$ has to transmit at least one message, which results in a lower bound message complexity of $\Omega(|G[u,k]| \cdot \log n)$ bits.

In summary, approaches like BFP-GG [18, 107] and GDBF [187, 188] are not reactive in the sense of the upper bound $\mathcal{O}(|\tau(G)[v,k]| \cdot \log n)$ (i.e., they are not $\mathcal{O}$-reactive), but are better than conventional, beacon-based approaches based on full k-hop knowledge. Except for worst-case scenarios, the number of bits transmitted during algorithm execution is not in the order of $\Omega(|G[u,k]| \cdot \log n)$, where u is the network node (proxy) responsible for local view topology construction of node v.

For characterization of this advantage over conventional approaches, and in analogy to the class of $\mathcal{O}$-reactive algorithms, the class of Ω-*reactive* algorithms is defined as follows.

Definition 5.5 (Ω-reactive topology control)**.** A local view topology control algorithm for a k-local topology mapping $\tau : \mathcal{C} \to \mathcal{D}$ is called Ω-*reactive* if the following applies. Let $G \in \mathcal{C}$ be any graph with n nodes. For any node $v \in \tau(G)$ there exists a node $u \in G$, such that the local view $\tau(G)[v]$ can be constructed, while the following holds:

- At algorithm start, nodes in G are unaware of their network neighborhoods (information from previous algorithm executions is assumed to be outdated and discarded).
- Node u starts the algorithm by a single local broadcast.

- The number of bits transmitted during algorithm execution is, except for the worst-case, not in the order of $\Omega(|G[u,k]| \cdot \log n)$.[3]

5.3 Classification of Existing Approaches

Next, the concepts of $\mathcal{O}$- and Ω-reactive topology control are employed on existing algorithms. The resulting taxonomy is provided in Table 5.1. Note that in all below-mentioned algorithms, nodes v and u as defined in Definitions 5.4 & 5.5 are the same. Therefore, only *executing node* v is mentioned.

Theorem 5.6. *BFP-GG [18] and GDBF [188] are Ω-reactive local view topology control algorithms.*

Proof. Recall the discussion of worst-case message complexities of algorithms BFP-GG and GDBF in Section 4.5, in particular the example unit disk graph $G = (V, E)$ which is given in Figure 4.11. Furthermore, recall the execution of algorithm BFP-GG or GDBF if initiated by node $v \in V$.

During algorithm execution, every node $w_i \in \mathrm{N}_1(v) \setminus \{v\}$ sends at least one message (either a CTS, or a PROTEST). Hence, in the worst-case, $\Theta(n)$ messages are transmitted, assuming that $|\mathrm{N}_1(v)| \in \Omega(n)$. On the contrary, $|\tau(G)[v]| = 1$. Even if each message is only of size $\mathcal{O}(1)$ bits, message complexity of BFP-GG and GDBF is $\Omega(n)$ bits.

On the other hand, it is easy to construct scenarios where the only Gabriel graph neighbor of executing node v suppresses CTS responses of arbitrarily many other neighbors in $\mathrm{N}_1(v)$. To see this, consider UDG $G' = (V', E')$ depicted in Figure 5.1, which is defined as follows.

Let $V' = \{v, w, u_1, ..., u_{n-2}\}$ be a set of nodes, such that any two nodes in V are at distance at most R from each other. Moreover, let V' be such that vw is an edge of the Gabriel graph, whereas for all $1 \leq i \leq n-2$ the Gabriel circle of any edge vu_i contains node w, which implies in particular that $\|vw\| < \|vu_i\|$ holds. Now, consider the execution of BFP-GG or GDBF by node v. Then, although $|G'[v]| = n-1$, only the single neighbor w of v in $\tau(G')[v]$ responds with a CTS and suppresses all other nodes' CTS responses. RTS and CTS messages contain at most a single node position/address and hence, any such message is of size at most $\mathcal{O}(\log n)$ bits. Therefore, $\Omega(|G[v]| \cdot \log n)$ is not a lower bound for the message complexity of BFP-GG and GDBF. □

Theorem 5.7. *BFP-RNG and BFP-CNG [18] are $\mathcal{O}$-reactive local view topology control algorithms.*

Proof. In Theorems 3 & 4 and Corollaries 2 & 3 in [18], Rührup et al. prove that during the execution of BFP-RNG and BFP-CNG at most a constant number of non-RNG (non-

[3] More formally, $\Omega(|G[u,k]| \cdot \log n)$ is an existential but not a universal lower bound for the message complexity of the algorithm [87]. I.e., for all sufficiently large positive values of n there exists an n-node graph $G' \in \mathcal{C}$, $v' \in \tau(G)$, and $u' \in G'$, s.t. the number of bits transmitted during algorithm execution by u' is not in the order of $\Omega(|G'[u',k]| \cdot \log n)$.

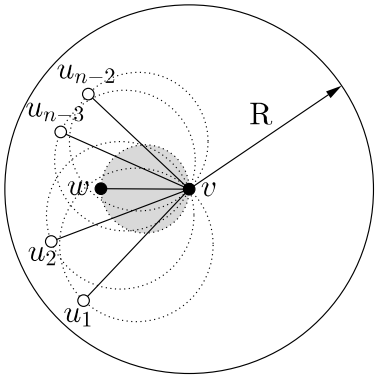


Figure 5.1: Example where GDBF and BFP-GG produce $\mathcal{O}(1)$ message transmissions during execution by node v. Hidden nodes are white and dotted circles represent GG-circles.

CNG) neighbors of executing node v send CTS messages.[4]Consequently, only a constant number of PROTEST messages is required in order to obtain the correct output graphs. Moreover, each such message contains at most a constant number of node positions. Hence, these algorithms have message complexities of $\mathcal{O}(|\tau(G)[v]| \cdot \log n)$ bits. □

Table 5.1 summarizes these results.

Algorithm	Topology	$\mathcal{O}$-reactive	Ω-reactive	References
GDBF [a]	GG		✓	[187, 188]
BFP-GG	GG		✓	[18, 107]
BFP-RNG	RNG	✓		[18, 107]
BFP-CNG	CNG	✓		[18, 107]

[a] GDBF refers here to the modified variant of the recovery strategy as described in Section 4.5

Table 5.1: Classification of existing approaches according to the definitions of $\mathcal{O}$- and Ω-reactive topology control.

5.4 Derivation of Fundamental Propositions

The concept of $\mathcal{O}$- and Ω-reactive local topology control does not merely enable classification and comparison of existing approaches. It is more powerful in the sense that it facilitates proofs of fundamental propositions, e.g., propositions of the type:

> "For graph class $\mathcal{C}$, there is no $\mathcal{O}$-reactive local topology control algorithm for construction of topology T."

[4]These proofs presuppose that the pairwise node distances in the input unit disk graph are distinct and therefore, the maximum node degrees of the output graphs are bounded by a constant.

In the following, two such propositions are proven regrading the reactive computability of the Gabriel graph and the *grid graph* [150]. Both the Gabriel and the grid graph are important and frequently used topology control structures. For the Gabriel graph the latter statement is beyond question considering the numerous approaches in the context of geographic routing and graph planarization presented in Chapter 3 that rely on local Gabriel graph construction. This is also true for grid graphs, which are used, e.g., in the local planarization algorithms [51, 70, 71, 138, 139, 149, 150] introduced in Chapter 3. That is, all algorithms relying on the reactive construction of these structures have generally to take into account that this imposes a severe message overhead, at least in worst-case scenarios.

Theorem 5.8. *For the class of unit disk graphs, there is no $\mathcal{O}$-reactive local view topology control algorithm for construction of the Gabriel graph neighborhood.*

Proof. The proof is based on the unit disk graph $G = (V, E)$ previously defined for the discussion of message complexities of BFP-GG and GDBF in Section 4.5 (see Figure 4.11 there for an illustration).

Let $G = (V, E)$ be defined as follows. $V = \{v, w_1, ..., w_{n-1}\}$, $\mathrm{D}(v, w_1) \cap V \setminus \{v, w_1\} = \emptyset$, $\forall\, 1 < i \leq n-1 : \mathrm{D}(v, w_i) \cap V \setminus \{v, w_i\} = w_{i-1}$, and $\forall\, u, v \in V$, $uv \in E$ holds.

Suppose there is a correct $\mathcal{O}$-reactive local view topology control algorithm $\mathcal{A}$ for construction of a node's local view on GG. Since $|\tau(G)[v]| = 1 \in \mathcal{O}(1)$ and $\mathcal{A}$ is $\mathcal{O}$-reactive, message complexity of execution of $\mathcal{A}$ by node v is bounded by $\mathcal{O}(\log n)$ bits. Assume that a single node position can be represented by $\mathcal{O}(1)$ bits. Then during $\mathcal{A}$'s execution, a subset of nodes $V' \subset V$ of size $|V'| \in \mathcal{O}(\log n)$ may reveal their positions, whereas all remaining $\Omega(n)$ nodes in $V \setminus V'$ are not allowed to do so. Then there must exist two consecutive nodes $w_l, w_{l+1} \in V \setminus V'$, where $1 < l < n-1$, that have not revealed their positions during algorithm execution.

Let G' be the unit disk graph obtained from G by removing w_l from V. Note that w_{l+1} is now a GG-neighbor of v in G' and it still holds that $|\tau(G)[v]| \in \mathcal{O}(1)$. Now consider the local views $G[w_{l+1}]$ and $G'[w_{l+1}]$ after the executions of $\mathcal{A}$ by node v w.r.t. both graphs G and G'. If w_{l+1} decides not to reveal its position during execution of $\mathcal{A}$ on G, then it will decide symmetrically during execution of $\mathcal{A}$ on G' since the local views $G[w_{l+1}]$ and $G'[w_{l+1}]$ are identical apart from knowledge about nodes that are irrelevant for the decision of whether vw_{l+1} is a Gabriel graph edge, or not. However, since v is not aware of its neighborhood at algorithm start, and w_{l+1} does not reveal its existence in both executions, the output of $\mathcal{A}$ cannot contain information about w_{l+1} and hence, algorithm $\mathcal{A}$ is not correct. □

Given a quasi unit disk graph $G = (V, E)$ with transmission radii r and R, define the *grid graph* $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ for G as follows (see Figure 5.2): Place an infinite, axis parallel square grid, with grid cell diameter r on the plane. For each non-empty grid cell, i.e., cells containing at least one node $v \in V$, create a new vertex c in $\mathcal{V}$ and map all nodes from V located in this cell to c. Any two vertices $c, z \in \mathcal{V}$ are connected by an undirected edge $e \in \mathcal{E}$, if and only if there are two nodes from V in the corresponding grid cells that are adjacent in G. Grid graph $\mathcal{H}$ is a connected overlay graph for G with various useful

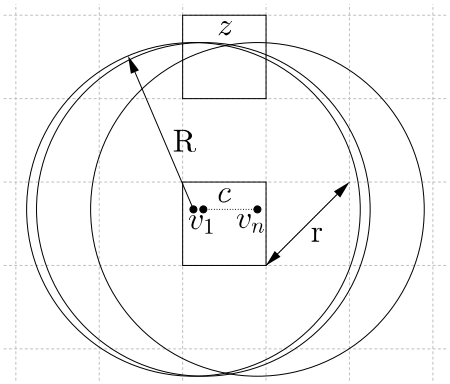


Figure 5.2: Illustration for the proof of Theorem 5.9. All nodes in graph $G = (V, E)$ are aligned in a single cell c and each node from V covers a unique part of cell z. Grid cells have diameter r. The circles represent the transmission radii of maximum transmission radius R.

properties (cf. [150, Theorem 1] and Chapter 8) and applications in local topology control and georouting as shown in Chapter 3. For a node $v \in V$, the local view $\tau(G)[v]$ is then the adjacency of its corresponding grid cell vertex in $\mathcal{H}$.

Theorem 5.9. *For the class of quasi unit disk graphs, there is no $\mathcal{O}$-reactive local view topology control algorithm for construction of a node's adjacency in the grid graph.*

Proof. Consider the quasi unit disk graph $G = (V, E)$ illustrated by Figure 5.2. Nodes in V are aligned axis-parallel and belong to one particular grid cell c. Then there exists a cell z, s.t. the transmission area $\mathrm{C_R}(v_i)$ of any node $v_i \in V$ covers a unique subarea of z. Note that $|\tau(G)[v_i]| = 0$, for all $v_i \in V$, and hence any $\mathcal{O}$-reactive local view topology control algorithm may exchange at most $\mathcal{O}(\log n)$ bits. Suppose there exists such an algorithm $\mathcal{A}$.

At algorithm start all nodes are unaware of their network neighbors. Each node is potentially the only one being connected in G to a node located in z. Therefore, each $v_i \in V$ has to transmit at least one message in order to provoke a response from a neighbor located in z. Even if each message is of size $\mathcal{O}(1)$ bits, algorithm $\mathcal{A}$ has a message complexity of $\Omega(n)$ bits in order to determine $\tau(G)[v_i]$ correctly, where v_i is an arbitrary node from V. This holds regardless of the choice of the node initiating algorithm execution. Hence, execution of $\mathcal{A}$ by any node v_i has a message complexity of at least $\Omega(n)$ bits.

Note that any "trivial" algorithm that simply outputs the empty set or tests only $\mathcal{O}(\log n)$ many nodes would fail, because one can always relocate any of the $\Omega(n)$ non-tested nodes $v_j \in V$ into the uniquely covered area of another non-tested node, such that this trivial algorithm would always fail to compute the correct output. □

5.5 Discussion

The concept of reactive local topology control enables profound research on the principal power of the reactive approach as well as an unambiguous classification of existing and prospective approaches in this field. It thereby supports identification of research gaps, open questions, and implications for existing and prospective work.

For example, the proof that generally no $\mathcal{O}$-reactive algorithm exists for construction of a node's local view on the Gabriel graph combined with the results given in Table 5.1 makes a rather surprising point: There are $\mathcal{O}$-reactive algorithms for construction of a node's local view on subgraphs of GG, such as RNG and CNG, and there is one for PDT which is a supergraph of GG (presented in Section 6.2). The resulting question is: Which structural property of GG hinders its $\mathcal{O}$-reactive computation?

Furthermore, the proof that generally no $\mathcal{O}$-reactive algorithm exists for construction of a node's adjacency in the grid graph can be used to show that the algorithm presented in Chapter 8 achieves an "optimal level" of reactivity.

It is worth noting that the above given definitions are held general enough in order to capture other future extensions, like virtual overlay graph constructions as proposed in [51, 70, 71, 136, 137, 150] as well as reactive approaches that are k-local and where $k > 1$.

Finally, the introduced concept is itself an object for future investigation. As yet only perfect localization and reliable communication have been considered. It is of particular interest to merge this deterministic concept with failure models describing the stochastic nature of localization and wireless communication.

Chapter 6

Reactive Local Construction of Euclidean Unit Disk Graph Spanners

When considering wireless networks modeled as unit disk graphs, the only reactive local topology control algorithms for construction of a node's local view on planar graphs are BFP-RNG, BFP-GG, BFP-CNG [18, 107], and GDBF [187, 188]. However, the graphs produced by these algorithms do not provide constant Euclidean spanning ratios, but have worst-case spanning ratios that depend on the total number of network nodes (see Section 3.1).

Until now in order to compute a node's local view on planar and constant stretch Euclidean spanners, it has been necessary to gather at least the node's one-hop neighborhood information and then to extract the desired local view, e.g., using the *partial unit Delaunay triangulation* (PuDel) [62].

In this chapter this research gap is closed. At first, in Section 6.1 it is proven that the *partial Delaunay triangulation* (PDT) [49, 50] constructed over a given unit disk graph is a connected, planar, and constant stretch Euclidean spanner for the input graph. Subsequently, in Section 6.2 the $\mathcal{O}$-reactive local topology control algorithm *reactivePDT* (rPDT for short) is introduced. Given a unit disk graph UDG(V), after execution of rPDT by any node $u \in V$, it outputs this node's local view on PDT(V), the partial Delaunay triangulation over node set V, using an optimal number of message transmissions.

These contributions are relevant: Firstly, the question whether PDT is a constant Euclidean spanner or not has long been an unanswered question (see Section 6.1.4 for details). Secondly, rPDT is the first $\mathcal{O}$-reactive local topology control algorithm for construction of constant stretch Euclidean spanners at all.

6.1 Euclidean Spanning Ratio of the Partial Delaunay Triangulation

In the following it is proven that the partial Delaunay triangulation (PDT) over a given unit disk graph is a Euclidean $(\frac{1+\sqrt{5}}{4}\pi^2)$-spanner of this unit disk graph. In order to prove this property, it is shown that PDT and the *partial unit Delaunay triangulation* (PuDel) [62] (see Definition 3.1 and Definition 3.5, respectively) yield equivalent graphs, if applied to the same unit disk graph. The desired result follows then from the fact that PuDel is a Euclidean $(\frac{1+\sqrt{5}}{4}\pi^2)$-spanner of the underlying unit disk graph, which was first

proven by Xu et al. in [62]. Since the constant spanning property of PuDel is crucial for proving the constant spanning property of PDT, after discussing a few preliminaries in Section 6.1.1, in Section 6.1.2 the proof that PuDel is a constant stretch spanner is revisited and completed. The proof of equivalence is then presented in Section 6.1.3. The matter of fact that PDT is a constant stretch Euclidean spanner for UDG has several implications. These are subsequently presented and discussed in Section 6.1.4.

6.1.1 Preliminaries

Model and Assumptions

Throughout this section, network graphs are modeled as unit disk graphs. $\mathrm{UDG}(V)$ always denotes the unit disk graph with unit radius R over a finite and distinct node set $V \subset \mathbb{R}^2$, which is further restricted as follows.

1. $3 \leq |V|$
2. No three points of V are collinear
3. No four points of V are cocircular
4. $\mathrm{UDG}(V)$ is a connected graph

Assumptions 1–3 ensure that Voronoi edges of the Voronoi diagram $\mathrm{VD}(V)$ are neither infinite straight lines, nor degenerate into a single point, which simplifies the proofs (for details, see Section 2.5.2). The results presented here could also be proven without these assumptions, but it would complicate the proofs without adding any additional insights. Note that the spanning ratio for two arbitrary nodes in an unconnected graph is undefined and therefore assumption 4 is required.

Notations

For convenience, the following notation is used repeatedly. It refers to the subset of nodes of V that are contained by $\mathrm{C_R}(x)$, the circle with radius R centered at x.

$$\mathrm{C_R}(x) \cap V = \{v \in V : \|xv\| \leq \mathrm{R}\} \subseteq V\,.$$

For formal definitions of the following terms, see Sections 2.5.2 and 3.1.

Given node set V and $u \in V$, $\mathrm{VR}_V(u)$ denotes the *Voronoi region* of u w.r.t. set V. $\mathrm{VD}(V)$ refers to the *Voronoi diagram* generated by V. The *Delaunay triangulation* of set V is denoted by $\mathrm{Del}(V)$. The *unit Delaunay triangulation* (UDel), the intersection of $\mathrm{Del}(V)$ and $\mathrm{UDG}(V)$, is denoted by $\mathrm{UDel}(V)$.

The *Gabriel graph* (GG) over $\mathrm{UDG}(V)$ is denoted by $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ to avoid confusion with the Gabriel graph which is defined over the complete Euclidean graph w.r.t. V. The *partial Delaunay triangulation* (PDT), as well as the *partial unit Delaunay triangulation* (PuDel) are only well defined over a given unit disk graph $\mathrm{UDG}(V)$ and are therefore simply denoted $\mathrm{PDT}(V)$ and $\mathrm{PuDel}(V)$, respectively. Recall that the definition of PuDel requires the notion of *directed local Delaunay edges* (see Definition 3.3) between two nodes u and v, which are denoted by $\overrightarrow{uv}$.

6.1.2 PuDel has Constant Euclidean Spanning Ratio

In the following it is shown that $\mathrm{PuDel}(V)$ is a constant stretch Euclidean spanner of the underlying unit disk graph $\mathrm{UDG}(V)$. This was first proven by Xu et al. in [62]. The proofs presented here complete and simplify the original proofs in [62].[1]

Proof Structure

The structure of the proof is as follows. At first, the auxiliary Lemmata 6.1–6.5 are proven. Subsequently, in Theorem 6.6, Corollaries 6.7 & 6.10, and Lemma 6.9, it is established that $\mathrm{GG}(V) \cap \mathrm{UDG}(V) \subseteq \mathrm{PuDel}(V) \subseteq \mathrm{UDel}(V)$. Theorem 6.11 and Corollary 6.12 show that $\mathrm{PuDel}(V)$ is symmetric and can be constructed based on one-hop neighborhood information. Finally, in showing that the direct DT path between any two nodes in $\mathrm{Del}(V)$ is a connected path in $\mathrm{PuDel}(V)$ (Theorem 6.18), the proof is given that $\mathrm{PuDel}(V)$ is a constant stretch Euclidean spanner of $\mathrm{UDG}(V)$ (Theorems 6.19 & 6.20).

Auxiliary Lemmata

Lemma 6.1. *For any $p \in \mathrm{VR}_V(u)$, $\mathrm{D}^{\circ}_{\|pu\|}(p)$ is empty of nodes from V.*

Proof. Note that $u \notin \mathrm{D}^{\circ}_{\|pu\|}(p)$. Moreover, by definition of a Voronoi region it holds that $\|pu\| \leq \|pv\|$, for all $v \in V \setminus \{u\}$, and hence $\mathrm{D}^{\circ}_{\|pu\|}(p)$ does also not contain any node from $V \setminus \{u\}$. □

Lemma 6.2. *For a point $p \notin \mathrm{H}(u,v)$, $\|pu\| > \|uv\|/2$ holds.*

Proof. By assumption p is not contained in $\mathrm{H}(u,v)$, i.e., $\|pu\| > \|pv\|$. By the triangle inequality it holds that $\|uv\| \leq \|pu\| + \|pv\| < 2 \cdot \|pu\|$ and hence, $\|uv\|/2 < \|pu\|$. □

Lemma 6.3 (Lemma 2.1 in [215])**.** *If the circle C expanding from a point $x \in \mathbb{R}^2$ hits exactly one node $p \in V$, then x belongs to $\mathrm{VR}_V(p)$. If C hits exactly two nodes, $p, q \in V$, then x is an interior point of a Voronoi edge separating the Voronoi regions of p and q. If C hits three or more nodes from V simultaneously, then x is a Voronoi vertex adjacent to those Voronoi regions whose nodes have been hit.*

Proof. See proof of Lemma 2.1 in [215]. □

Lemma 6.4. $\mathrm{VR}_V(u) \cap \mathrm{B}(u,v) = \mathrm{VR}_V(v) \cap \mathrm{B}(u,v)$.

Proof. In the following it is shown that $\mathrm{VR}_V(u) \cap \mathrm{B}(u,v) \subseteq \mathrm{VR}_V(v) \cap \mathrm{B}(u,v)$. The remaining case can be shown analogously.

If $\mathrm{VR}_V(u) \cap \mathrm{B}(u,v) = \emptyset$, then for any point $p \in \mathrm{B}(u,v)$ it holds that there exists a node $x \in V$ with $\|px\| < \|pu\|$. Since $p \in \mathrm{B}(u,v)$, it follows that $\|px\| < \|pv\|$ and thus, no such p is contained in $\mathrm{VR}_V(v)$. Hence, $\mathrm{VR}_V(v) \cap \mathrm{B}(u,v) = \emptyset$ holds.

[1] The author of this thesis would like to emphasize that the proof of Lemma 6.5 is identical to the one given in [62]. The proofs of Theorems 6.11, 6.18, and 6.19 as well as of Corollary 6.12 are given with minor revisions. All other proofs concerning the spanning ratio of PuDel either include major revisions or are completely new.

If $\mathrm{VR}_V(u) \cap \mathrm{B}(u,v) \neq \emptyset$, consider any point p from $\mathrm{VR}_V(u) \cap \mathrm{B}(u,v)$. By Lemma 6.1 $\mathrm{D}^{\circ}_{\|up\|}(p)$ is empty of nodes from V. Since $p \in \mathrm{B}(u,v)$, both u and v are on the boundary of $\mathrm{C}_{\|up\|}(p)$. The application of Lemma 6.3 yields that $p \in \mathrm{VR}_V(v)$ and hence, it holds that $p \in \mathrm{VR}_V(v) \cap \mathrm{B}(u,v)$. □

Lemma 6.5. *For any node $u \in V$ it holds that $\mathrm{VR}_V(u) \subseteq \mathrm{VR}_{\mathrm{N}_1(u)}(u)$.*

Proof.

$$\begin{aligned}
\mathrm{VR}_V(u) &= \bigcap_{v \in V \setminus \{u\}} \mathrm{H}(u,v) \\
&= \left(\bigcap_{v \in \mathrm{N}_1(u) \setminus \{u\}} \mathrm{H}(u,v) \right) \cap \left(\bigcap_{v \in V \setminus \mathrm{N}_1(u)} \mathrm{H}(u,v) \right) \\
&= \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \left(\bigcap_{v \in V \setminus \mathrm{N}_1(u)} \mathrm{H}(u,v) \right)
\end{aligned}$$

□

Properties of PuDel

Theorem 6.6. *Let $u, v \in V$. Segment $\overline{uv}$ is a locally detectable directed local Delaunay edge $\overrightarrow{uv}$ if and only if $uv \in \mathrm{UDel}(V)$ and there exists $p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ with $\|up\| \leq \mathrm{R}/2$.*

Proof. "⇒": Let $\overrightarrow{uv}$ be a locally detectable directed local Delaunay edge. Then the following properties hold:

1. $\|uv\| \leq \mathrm{R}$ and $uv \in \mathrm{Del}(\mathrm{N}_1(u))$ (by definition of a directed local Delaunay edge).
2. $\exists\, p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ with $\|up\| \leq \mathrm{R}/2$ (by definition of a locally detectable directed local Delaunay edge), and
3. $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{B}(u,v)$ (follows from 2.).

By Lemma 6.5 it holds that $\mathrm{VR}_V(u) \subseteq \mathrm{VR}_{\mathrm{N}_1(u)}(u)$. Distinguish the Cases (i) and (ii).

Case (i): Assume $p \notin \mathrm{VR}_V(u)$. Then there exists $l \in V \setminus \mathrm{N}_1(u)$ s.t. $p \notin \mathrm{H}(u,l)$. By Lemma 6.2 it holds that $\|up\| > \|ul\|/2$. Because $l \notin \mathrm{N}_1(u)$, it holds that $\|ul\| > \mathrm{R}$. This implies that $\|up\| > \|ul\|/2 > \mathrm{R}/2$. But this is a contradiction to the assumption that $\|up\| \leq \mathrm{R}/2$ and therefore, only Case (ii) may actually apply.

Case (ii): Assume $p \in \mathrm{VR}_V(u)$. By property 3., $p \in \mathrm{B}(u,v)$ and by Lemma 6.4 it holds that $p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$, which implies that $uv \in \mathrm{Del}(V)$. With properties 1. and 2., $\|uv\| \leq \mathrm{R}$ and $\|up\| \leq \mathrm{R}/2$ holds, which proves the first part of this claim.

"⇐": Assume $uv \in \mathrm{UDel}(V)$. Then, by definition of the Delaunay triangulation $\mathrm{VR}_V(u) \cap \mathrm{VR}_V(v) \neq \emptyset$ and it holds that $\|uv\| \leq \mathrm{R}$. Now assume $\exists\, p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ with $\|up\| \leq \mathrm{R}/2$. Note that $p \in \mathrm{B}(u,v)$ and hence, $p \in \mathrm{VR}_V(u) \cap \mathrm{B}(u,v)$. By Lemma 6.5,

it holds that $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u)$ and thus, $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{B}(u,v)$. Lemma 6.4 implies that $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(v)$. Therefore, uv is a locally detectable directed local Delaunay edge. □

Corollary 6.7. $\mathrm{PuDel}(V)$ *is a subgraph of* $\mathrm{UDel}(V)$ *and a planar graph.*

Proof. Theorem 6.6 implies that $\mathrm{PuDel}(V)$ is a subgraph of $\mathrm{UDel}(V)$. Since $\mathrm{UDel}(V)$ is a subgraph of the planar graph $\mathrm{Del}(V)$, $\mathrm{PuDel}(V)$ is also a planar graph. □

The proof that $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ is a subgraph of $\mathrm{PuDel}(V)$ requires the following observation.

Lemma 6.8 (Theorem 7.4 in [216])**.** *The following holds for the Voronoi diagram* $\mathrm{VD}(V)$*:*

(i) A point $q \in \mathbb{R}^2$ is a Voronoi vertex in $\mathrm{VD}(V)$*, if and only if its* largest empty circle $C_V(q)$ *(the largest circle centered at q which does not contain any node from V) contains three or more nodes from V on its boundary.*

(ii) The bisector between nodes $u, v \in V$ defines a Voronoi edge of $\mathrm{VD}(V)$*, if and only if there is a point q on the bisector such that the largest empty circle $C_V(q)$ contains both u and v on its boundary but no other node from V.*

Proof. See proof of Theorem 7.4 in [216]. □

Lemma 6.9. $\mathrm{GG}(V) \cap \mathrm{UDG}(V) \subseteq \mathrm{PuDel}(V)$.

Proof. Let uv be any edge in $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$. In the remainder it is shown that $uv \in \mathrm{PuDel}(V)$.

Let m denote the midpoint of $\overline{uv}$, for which it holds that $m \in \mathrm{B}(u,v)$. By assumption $\|uv\| \leq \mathrm{R}$ and since uv is a Gabriel graph edge, there are no nodes from $V \setminus \{u,v\}$ contained in the circle $\mathrm{C}_{\|um\|}(m)$. By Lemma 6.8(ii), m is a point on the Voronoi edge between u and v w.r.t. V, i.e., $m \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$. By Lemma 6.5, $m \in \mathrm{VR}_{\mathrm{N}_1(u)}(u)$ and by Lemma 6.4 it holds that $m \in \mathrm{VR}_{\mathrm{N}_1(u)}(v)$. Thus, $m \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ and $\|um\| \leq \mathrm{R}/2$. Hence, uv is a locally detectable directed local Delaunay edge and therefore it belongs to $\mathrm{PuDel}(V)$. □

Corollary 6.10. *If* $\mathrm{UDG}(V)$ *is a connected graph, then* $\mathrm{PuDel}(V)$ *is a connected graph.*

Proof. Bose et al. prove in Lemma 1 in [36] that $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ is connected if $\mathrm{UDG}(V)$ is connected. By Lemma 6.9, $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ is a subgraph of $\mathrm{PuDel}(V)$ and therefore it holds that $\mathrm{PuDel}(V)$ is also connected if $\mathrm{UDG}(V)$ is connected. □

Theorem 6.11. $\mathrm{PuDel}(V)$ *is a symmetric graph, i.e., if $\overrightarrow{uv}$ is a locally detectable directed local Delaunay edge, then $\overrightarrow{vu}$ is also a locally detectable directed local Delaunay edge.*

Proof. Let $u, v \in V$ s.t. $\overrightarrow{uv}$ is a locally detectable directed local Delaunay edge. The following proof shows that from the view point of v, $\overline{vu}$ is also a locally detectable directed local Delaunay edge. That is, there exists $p \in \mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)$ with $\|vp\| \leq \mathrm{R}/2$ and $\|vu\| \leq \mathrm{R}$.

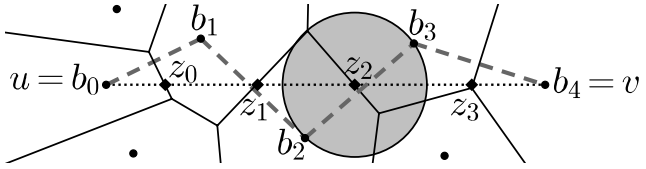


Figure 6.1: Illustration of the direct DT path $\mathrm{DT}(u, v) = \langle u = b_0, b_1, b_3, b_4 = v \rangle$, which is represented by the dashed line. The solid lines represent $\mathrm{VD}(V)$, whereas $\overline{uv}$ is given by the dotted line.

By Theorem 6.6, $uv \in \mathrm{UDel}(V)$ and $\exists\, p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ with $\|up\| \leq \mathrm{R}/2$. Thus, $p \in \mathrm{B}(u, v)$ and by definition of directed local Delaunay edges it also holds that $\|uv\| \leq \mathrm{R}$. Lemma 6.5 yields that $\mathrm{VR}_V(v) \subseteq \mathrm{VR}_{\mathrm{N}_1(v)}(v)$ and thus, $p \in \mathrm{VR}_{\mathrm{N}_1(v)}(v)$. Because $\|up\| \leq \mathrm{R}/2$ and $p \in \mathrm{B}(u, v)$ it also holds that $\|vp\| \leq \mathrm{R}/2$. In combination with Lemma 6.4, from this it can be concluded that $\overline{vu}$ is a locally detectable directed local Delaunay edge. □

The previous result almost immediately implies the following corollary.

Corollary 6.12. *The partial unit Delaunay graph for the set V, $\mathrm{PuDel}(V)$, can be computed locally using only one-hop neighborhood information.*

Proof. Each node $v \in V$ queries its one-hop neighbors for their positions and computes the local Voronoi diagram $\mathrm{VD}(\mathrm{N}_1(v))$. Based upon this diagram, each node decides locally which of its incident edges are locally detectable directed local Delaunay edges. From Theorem 6.11 it follows that an edge uv added by node u to $\mathrm{PuDel}(V)$ is also added to this graph by node v based on its one-hop neighborhood information. □

Euclidean Spanning Ratio of PuDel

The proof of the constant Euclidean spanning ratio of PuDel requires the well-known concept of *DT-paths* introduced by Dobkin et al. in their seminal work [101].

Definition 6.13 (Direct DT path [101])**.** Let $u, v \in V$ and assume for simplicity that $\overline{uv}$ is parallel to the x-axis and that $\mathrm{x}(u) < \mathrm{x}(v)$. The *direct DT path* between u and v, denoted by $\mathrm{DT}(u, v)$, is a sequence $\langle u = b_0, b_1, ..., b_{m-1}, b_m = v \rangle$ of nodes from V that corresponds to the sequence of Voronoi regions traversed by walking from u to v along the line segment $\overline{uv}$ (see Figure 6.1 for an illustration). In case a Voronoi edge lies on $\overline{uv}$, the Voronoi region which lies above $\overline{uv}$ is chosen.

If all nodes b_i along the direct DT path happen to be in the same half-plane defined by the line connecting u and v, then this path is said to be *one-sided*.

Let $\mathrm{DT}(u, v) = \langle u = b_0, b_1, ..., b_{m-1}, b_m = v \rangle$ be the direct DT path between two nodes $u, v \in V$. The following three lemmata are proven by Dobkin et al. in [101].

Lemma 6.14 (Lemma 1 in [101])**.**

$$\mathrm{x}(b_0) \leq \mathrm{x}(b_1) \leq ... \leq \mathrm{x}(b_m)\,.$$

Lemma 6.15 (Lemma 2 in [101]). *$\forall\, i,\ 0 \leq i \leq m$, b_i is contained by* $\mathrm{D}(u, v)$.

Lemma 6.16 (Lemma 3 in [101]). *Let $D_1, D_2, ..., D_k$ be circles which are centered at the* x*-axis, such that $D = \bigcup_{1 \leq i \leq k} D_i$ is connected. Then the boundary of D has a length of at most $\pi \cdot (x_r - x_l)$, where x_l and x_r are the least and the greatest* x*-coordinates of D, respectively.*

Lemma 6.17. *Let $u, v \in V$ s.t. the direct DT path from u to v is one-sided, then the length of this path is at most $(\pi/2) \cdot \|uv\|$.*

Proof. Let $\mathrm{DT}(u, v) = \langle u = b_0, b_1, ..., b_m = v \rangle$ be a one-sided direct DT path between $u, v \in V$. By Lemma 6.14, $\mathrm{x}(b_0) \leq \mathrm{x}(b_1) \leq ... \leq \mathrm{x}(b_m)$ holds. For all i, $0 \leq i < m$, the Euclidean length of edge $b_i b_{i+1}$ is bounded from above by the length of the circular arc between b_i and b_{i+1} w.r.t. the circle $\mathrm{C}_{\|z_i b_i\|}(z_i)$, where z_i is the point on the x-axis where the Voronoi regions of b_i and b_{i+1} intersect. Hence, the Euclidean length of $\mathrm{DT}(u, v)$ is upper bounded by the sum of the respective arcs. By Lemma 6.16, the length of the connected sequence of arcs is bounded from above by $(\pi/2) \cdot \|uv\|$. □

Theorem 6.18. *If $uv \in \mathrm{UDel}(V)$, then the direct DT path from u to v in $\mathrm{Del}(V)$ is a connected path in $\mathrm{PuDel}(V)$.*

Proof. Let $uv \in \mathrm{UDel}(V)$ be any edge and $\mathrm{DT}(u, v) = \langle u = b_0, b_1, ..., b_{m-1}, b_m = v \rangle$ the corresponding direct DT path in $\mathrm{Del}(V)$. Let z_i, $0 \leq i < m$, be the point of intersection of the Voronoi edge between the nodes b_i, b_{i+1} (w.r.t. V) with the line segment $\overline{uv}$; i.e., $z_i \in \mathrm{VR}_V(b_i) \cap \mathrm{VR}_V(b_{i+1})$, $z_i \in \mathrm{B}(b_i, b_{i+1})$, and $z_i \in \overline{uv}$ (see Figure 6.1 for an illustration).

Since $z_i \in \mathrm{VR}_V(b_i)$, by Lemma 6.1 it holds that there are no points from V contained by the open disk $\mathrm{D}^{\circ}_{\|z_i b_i\|}(z_i)$, and this holds in particular for u and v. Observe that the circle's center, namely z_i, as well as the circle's diameter, are located on the line segment $\overline{uv}$. By assumption that $uv \in \mathrm{UDel}(V)$, it holds that $\|uv\| \leq \mathrm{R}$. It follows that $2 \cdot \|z_i b_i\| \leq \|uv\| \leq \mathrm{R}$. Since $z_i \in \mathrm{B}(b_i, b_{i+1})$, the triangle inequality yields

$$\|b_i b_{i+1}\| \leq \|b_i z_i\| + \|z_i b_{i+1}\| = 2 \cdot \|z_i b_i\| \leq \mathrm{R}\,.$$

Thus, $\|z_i b_i\| \leq \mathrm{R}/2$ and z_i is a point of the Voronoi edge between b_i and b_{i+1}. From Theorem 6.6 it now follows that $b_i b_{i+1} \in \mathrm{PuDel}(V)$. This holds for every edge $b_j b_{j+1}$, $0 \leq j < m$, along the path $\mathrm{DT}(u, v)$. Therefore, $\mathrm{DT}(u, v)$ is a connected path from u to v in $\mathrm{PuDel}(V)$. □

In the following, let $\Pi(u, v)$ denote a connected and acyclic path from u to v whose Euclidean length is given by $||\Pi(u, v)||$ (see Definition 2.11).

Theorem 6.19. *For each edge $uv \in \mathrm{UDel}(V)$ there exists a connected path $\Pi(u, v)$ in $\mathrm{PuDel}(V)$ of Euclidean length $||\Pi(u, v)|| \leq (\pi/2) \cdot \|uv\|$.*

Proof. Let $uv \in \mathrm{UDel}(V)$ be an arbitrarily chosen but fixed edge. Distinguish the Cases (i) and (ii).

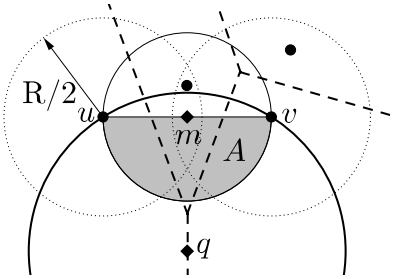


Figure 6.2: Illustration for the proof of Theorem 6.19, Case (ii). The shaded area represents the lower half-circle of $\mathrm{D}(u, v)$. $\mathrm{C}_{\|uq\|}(q)$ is given by the bold black circle and $\mathrm{VD}(V)$ is represented by the dashed black lines.

Case (i): $\exists p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ with $\|up\| \leq \mathrm{R}/2$. In this case it holds that $uv \in \mathrm{PuDel}(V)$ according to Theorem 6.6. The path $\Pi(u, v) := \langle u, v \rangle$ is a connected path in $\mathrm{PuDel}(V)$ of Euclidean length

$$||\Pi(u, v)|| = \|uv\| < \frac{\pi}{2} \cdot \|uv\|.$$

Case (ii): $\nexists p \in \mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ with $\|up\| \leq \mathrm{R}/2$. Let q be any point in $\mathrm{VR}_V(u) \cap \mathrm{VR}_V(v)$ (see Figure 6.2 for an illustration). Because $q \in \mathrm{VR}_V(u)$ and $q \in \mathrm{VR}_V(v)$, it holds that $q \in \mathrm{VR}_V(u) \cap \mathrm{B}(u, v)$. From this and the assumption that $\|qu\| > \mathrm{R}/2$, it follows that $\|uq\| = \|vq\| > \mathrm{R}/2$.

Denote the midpoint of the line segment $\overline{uv}$ by m. Since $uv \in \mathrm{UDel}(V)$, $\|uv\| \leq \mathrm{R}$ and $\|um\| = \|vm\| \leq \mathrm{R}/2 < \|uq\|$. In particular, this implies that $m \neq q$. Now consider the following two circles with centers m and q, respectively, both of which contain u and v on their boundaries:

- $\mathrm{D}(u, v)$ represents the circle with center m, radius $\|um\|$, and which has line segment $\overline{uv}$ as its diameter.
- $\mathrm{C}_{\|uq\|}(q)$ is the circle with center q, radius $\|uq\|$, and which contains both u and v on its boundary, since $\|uq\| = \|vq\|$.

Note that these circles do not coincide, as their centers as well as their radii differ. Also, observe that one of the semicircles of $\mathrm{D}(u, v)$ that is bounded by $\overline{uv}$, which is henceforth called A (see shaded area in Figure 6.2), is entirely contained by $\mathrm{C}_{\|uq\|}(q)$. Since $q \in \mathrm{VR}_V(u)$, by Lemma 6.1 it holds that no points from V are contained in the open disk $\mathrm{D}^{\circ}_{\|uq\|}(q)$. From this, it can be concluded that no points from $V \setminus \{u, v\}$ are contained inside or on the boundary of semicircle A.

Let $\mathrm{DT}(u, v)$ denote the direct DT path from u to v in $\mathrm{Del}(V)$. By Lemma 6.15 it holds that all nodes along the path $\mathrm{DT}(u, v)$ are contained within or on the boundary of $\mathrm{D}(u, v)$. But since semicircle A of $\mathrm{D}(u, v)$ is empty of nodes from $V \setminus \{u, v\}$, all remaining nodes from $\mathrm{DT}(u, v)$ must be contained in the remaining semicircle $\mathrm{D}(u, v) - A$; i.e., all nodes from $\mathrm{DT}(u, v)$ are contained in the closed half-plane that has $\overline{uv}$ as its boundary

and contains $\mathrm{D}(u,v) - A$. Thus, $\mathrm{DT}(u,v)$ is one-sided. By Theorem 6.18, $\mathrm{DT}(u,v)$ is a connected path in $\mathrm{PuDel}(V)$. Therefore, for $\Pi(u,v) := \mathrm{DT}(u,v)$ it holds by Lemma 6.17 that

$$||\Pi(u,v)|| \leq \frac{\pi}{2} \cdot \|uv\| .$$

□

This section now concludes with the proof that PuDel is a constant stretch Euclidean spanner of the unit disk graph. The spanning ratio proven here is slightly weaker than the one in the original proof [62]. They claim a spanning ratio of $(2\pi^2)/(3\sqrt{3})$, but their proof relies on an unproven claim concerning the spanning ratio of the unit Delaunay triangulation (for details see Section 2.5.2).

Theorem 6.20. *Let V be a set of nodes such that* $\mathrm{UDG}(V)$ *is a connected graph. Then,* $\mathrm{PuDel}(V)$ *is a Euclidean* $(\frac{1+\sqrt{5}}{4}\pi^2)$*-spanner of* $\mathrm{UDG}(V)$.

Proof. Let V be a set of nodes such that $\mathrm{UDG}(V)$ is connected. According to Theorem 5 in [104], $\mathrm{UDel}(V)$ is a Euclidean $(\frac{1+\sqrt{5}}{2}\pi)$-spanner of $\mathrm{UDG}(V)$. Next it is shown that $\mathrm{PuDel}(V)$ is a Euclidean $\pi/2$-spanner of $\mathrm{UDel}(V)$ from which the theorem then follows.

Let $u, v \in V$ be two arbitrarily chosen but fixed nodes. Assume the Euclidean shortest path from u to v in $\mathrm{UDel}(V)$ is given by

$$\Pi_{\mathrm{UDel}(V)}(u,v) = \langle u = u_0, u_1, ..., u_{k-1}, u_k = v \rangle , k \geq 1 .$$

By Theorem 6.19, for each edge $xy \in \mathrm{UDel}(V)$ there exists a connected path from x to y in $\mathrm{PuDel}(V)$ whose Euclidean length is at most $(\pi/2) \cdot \|xy\|$. In the following, denote this path by $x \rightsquigarrow y$. Each pair (u_i, u_{i+1}), for $0 \leq i < k$, represents an edge in $\mathrm{UDel}(V)$. Thus, the concatenation of all such paths $u_i \rightsquigarrow u_{i+1}$, for $0 \leq i < k$, is a connected path from u to v in $\mathrm{PuDel}(V)$.

Let $\Pi_{\mathrm{PuDel}(V)}(u,v)$ denote the Euclidean shortest path from u to v in $\mathrm{PuDel}(V)$. The Euclidean length of $\Pi_{\mathrm{PuDel}(V)}(u,v)$ can be upper bounded as follows.

$$\begin{aligned} ||\Pi_{\mathrm{PuDel}(V)}(u,v)|| &\leq \sum_{i=0}^{k-1} ||u_i \rightsquigarrow u_{i+1}|| \\ &\overset{\text{Thm. 6.19}}{\leq} \sum_{i=0}^{k-1} \frac{\pi}{2} \cdot \|u_i u_{i+1}\| = \frac{\pi}{2} \cdot \sum_{i=0}^{k-1} \|u_i u_{i+1}\| = \frac{\pi}{2} \cdot ||\Pi_{\mathrm{UDel}(V)}(u,v)|| . \end{aligned}$$

□

6.1.3 Equivalence of PuDel and PDT

It is now shown that the partial unit Delaunay triangulation (PuDel) and the partial Delaunay Triangulation (PDT) are equivalent. The main Theorem 6.27 requires few auxiliary Lemmata 6.21 –6.26 which are given first.

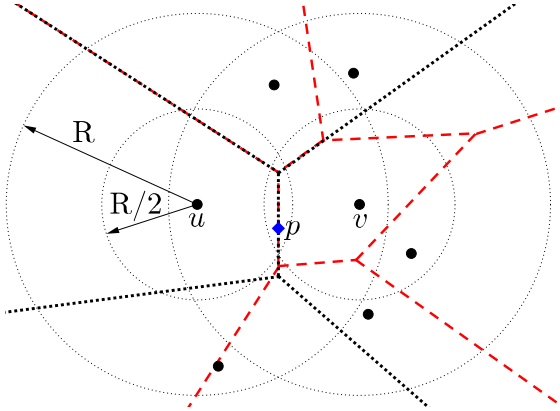


Figure 6.3: Illustration for the proof of Lemma 6.21. $\mathrm{VD}(\mathrm{N}_1(u))$ and $\mathrm{VD}(\mathrm{N}_1(v))$ are represented by dotted and dashed lines, respectively.

Lemma 6.21. *Let $uv \in \mathrm{PuDel}(V)$ and let $p \in \mathbb{R}^2$ be any* witness *for the local detectability of $\overrightarrow{uv}$ (i.e., $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ and $\|up\| \leq \mathrm{R}/2$). It holds that p is also a witness for the local detectability of $\overrightarrow{vu}$.*

Proof. Assume p is a witness for the local detectability of $\overrightarrow{uv}$, then $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ and $\|up\| = \|vp\| \leq \mathrm{R}/2$ (see Figure 6.3 for an illustration). In particular it holds that

$$p \in \bigcap_{x \in \mathrm{N}_1(u) \setminus \{u\}} \mathrm{H}(u, x)\,.$$

Assume, for the sake of contradiction, that $p \notin \mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)$. Then there must exist $l \in \mathrm{N}_1(v) \setminus \{u, v\}$ such that $p \notin \mathrm{H}(v, l)$ but $p \in \mathrm{H}(l, v)$. The latter implies that $\|lp\| < \|vp\|$. But then, l must be a one-hop neighbor of u for the following reason:

$$\|ul\| \leq \|up\| + \|lp\| < \|up\| + \|vp\| \leq \mathrm{R}\,,$$

where the first inequality holds due to the triangle inequality. Because $\|lp\| < \|vp\| = \|up\|$ it holds that $p \notin \mathrm{H}(u, l)$. But this contradicts the initial assumption that p is a witness for the local detectability of $\overrightarrow{uv}$ and hence, $p \in \mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)$ is a witness for the local detectability of $\overrightarrow{vu}$ as well. □

Definition 6.22 (Common local Voronoi edge)**.** For two nodes $u, v \in V$ that are connected by an edge in $\mathrm{UDG}(V)$, define the *common local Voronoi edge* of u and v w.r.t. their one-hop neighborhoods, denoted $\mathrm{CLVE}(u, v)$, as follows:

$$\mathrm{CLVE}(u, v) = \left\{ \left(\mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)\right) \cap \left(\mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)\right) \right\}\,.$$

For an example, see Figure 6.4 where $\mathrm{CLVE}(u, v)$ is represented by the line segment $\overline{ab}$.

Lemma 6.23. *Let $u, v \in V$. If $\mathrm{CLVE}(u, v) \neq \emptyset$, then $|\mathrm{CLVE}(u, v)| > 1$.*

Proof. For the sake of contradiction, assume there are two nodes $u, v \in V$ such that $|\mathrm{CLVE}(u,v)| = 1$, i.e., it consists of a single point. Distinguish the following two cases:

(i) $|\mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)| = 1$ or $|\mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)| = 1$;

(ii) $|\mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)| > 1$ and $|\mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)| > 1$.

Case (i): At least one of the two local Voronoi diagrams is *degenerated* since a Voronoi edge can only degenerate into a single point if four or more nodes are cocircular. This contradicts the assumption that no four points in V are cocircular. Hence, only the other case may actually apply.

Case (ii): Let p be the single point contained in $\mathrm{CLVE}(u,v)$. Observe that p must be a Voronoi vertex in both $\mathrm{VD}(\mathrm{N}_1(u))$ and $\mathrm{VD}(\mathrm{N}_1(v))$ (otherwise the Voronoi regions of u and v w.r.t. $\mathrm{N}_1(u)$ and $\mathrm{N}_1(v)$, respectively, would overlap and the size of $\mathrm{CLVE}(u,v)$ would be strictly larger than one). Assume p is the point of intersection of

- $\mathrm{VR}_{\mathrm{N}_1(u)}(u)$, $\mathrm{VR}_{\mathrm{N}_1(u)}(v)$, and $\mathrm{VR}_{\mathrm{N}_1(u)}(l)$, $l \in \mathrm{N}_1(u) \setminus \{u, v\}$, as well as of
- $\mathrm{VR}_{\mathrm{N}_1(v)}(u)$, $\mathrm{VR}_{\mathrm{N}_1(v)}(v)$, and $\mathrm{VR}_{\mathrm{N}_1(v)}(\hat{l})$, $\hat{l} \in \mathrm{N}_1(v)$.

Because the local Voronoi edges are assumed to be non-degenerated (following from the non-collinearity assumption) and because p is the only point of intersection of the local Voronoi edges, it holds that $l \neq \hat{l}$. But this implies that p is equidistant to u, v, l, and $\hat{l}$. This contradicts the assumption that no four points in V are cocircular. □

Lemma 6.24. *Let $uv \in \mathrm{PuDel}(V)$ be any edge with midpoint m. If there exists a closed interval $[a,b] \in \mathrm{CLVE}(u,v)$, such that $m \in [a,b]$ and $m \neq a, b$, then it holds that $\mathrm{D}(u,v)$ is empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u,v\}$.*

Proof. Let $uv \in \mathrm{PuDel}(V)$ be any edge and assume there exists a closed interval $[a,b] \in \mathrm{CLVE}(u,v)$, such that $m \in [a,b]$ and $m \neq a, b$.

Because Voronoi edges are continuous, w.l.o.g. it can be assumed that a and b are both contained in $\mathrm{C}_{\mathrm{R}/2}(u) \cap \mathrm{C}_{\mathrm{R}/2}(v)$. It may also be assumed w.l.o.g. that $\overline{uv}$ is parallel to the x-axis and that $\mathrm{y}(a) > \mathrm{y}(m)$. See Figure 6.4 for an illustration.

Next it is shown that the upper half-circle of $\mathrm{D}(u,v)$ containing a, denoted by A (see shaded area in Figure 6.4), is empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u,v\}$. In order to show that the remaining, lower half-circle is also empty, an analogue line of argumentation can be applied.

First of all, note that the radius of $\mathrm{D}(u,v)$ is strictly smaller than the radius of circle $\mathrm{C}_{\|ua\|}(a)$ (circle $\mathrm{C}(u,v,l)$ in Figure 6.4), which is due to the facts that m is the midpoint of uv, and m as well as a are distinct points on $\mathrm{B}(u,v)$. From this and from the assumption that $\mathrm{y}(a) > \mathrm{y}(m)$, it follows that semicircle A is entirely contained by $\mathrm{C}_{\|ua\|}(a)$. Because (i) $m, a \in \mathrm{B}(u,v)$ and (ii) $a \in \mathrm{CLVE}(u,v)$ it holds that

$$\|ua\| \overset{\text{(i)}}{=} \|va\| \overset{\text{(ii)}}{\leq} \|la\|, \forall\, l \in (\mathrm{N}_1(u) \cup \mathrm{N}_1(v))\,.$$

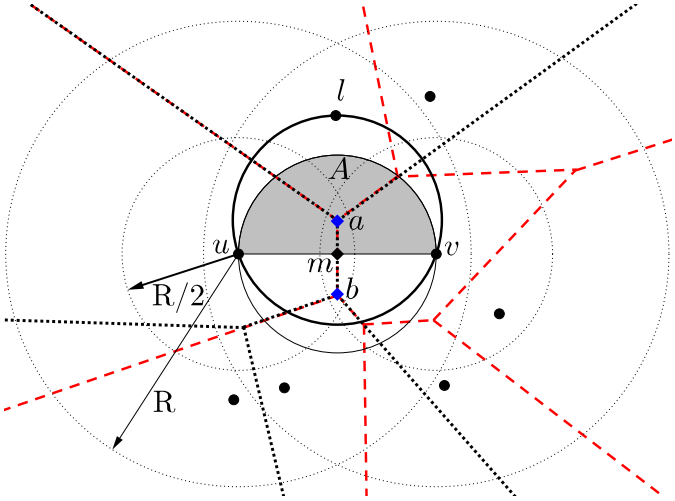


Figure 6.4: Illustration for the proof of Lemma 6.24. $\mathrm{CLVE}(u, v)$ is given by the line segment $\overline{ab}$, $\mathrm{VD}(\mathrm{N}_1(u))$ and $\mathrm{VD}(\mathrm{N}_1(v))$ are represented by the dotted and dashed lines, respectively. The shaded area represents the upper half-circle of $\mathrm{D}(u, v)$, whereas the bold black circle is $\mathrm{C}(u, v, l)$.

That is, the open disk $\mathrm{D}^{\circ}_{\|ua\|}(a)$ is empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u, v\}$ and hence, this also holds for A. The analogue line of argumentation regarding the other interval boundary point b completes this proof. □

Lemma 6.25 (Theorem 5.7 in [108])**.** *Given a point set V where no four points are cocircular and no three points are collinear, then every Voronoi vertex in $\mathrm{VD}(V)$ is the common intersection of exactly three edges of the diagram.*

Proof. See proof of Theorem 5.7 in [108]. □

Lemma 6.26 (Theorem 5.8 in [108])**.** *For every Voronoi vertex x of the Voronoi diagram $\mathrm{VD}(V)$, the circumcircle of its three generating nodes from V does not contain other nodes from V.*

Proof. See proof of Theorem 5.8 in [108]. □

Next the equivalence of PuDel and PDT is proven.

Theorem 6.27. *Let $V \subset \mathbb{R}^2$ be a distinct set of nodes of size $3 \leq |V| < \infty$, which satisfies that no four nodes are cocircular as well as that no three nodes are collinear. Then, for any two nodes $u, v \in V$ it holds that $uv \in \mathrm{PuDel}(V)$, if and only if $uv \in \mathrm{PDT}(V)$.*

Proof. This proof of equivalence is given in two parts:

Part (1): $uv \in \mathrm{PuDel}(V) \Rightarrow uv \in \mathrm{PDT}(V)$, and

Part (2): $uv \in \mathrm{PDT}(V) \Rightarrow uv \in \mathrm{PuDel}(V)$.

In the remainder of this proof it is assumed w.l.o.g. that line segment $\overline{uv}$ is parallel to the x-axis. Let m always denote the midpoint of $\overline{uv}$ and R the radius of the underlying unit disk graph.

In addition, note the following. If $uv \in \mathrm{UDG}(V)$, then $\mathrm{D}(u,v)$ is entirely contained by the unit disk centered at u. Thus, $\mathrm{D}(u,v) \cap (\mathrm{N}_1(u) \setminus \{u,v\}) = \emptyset$, if and only if $\mathrm{D}(u,v) \cap (V \setminus \{u,v\}) = \emptyset$.

Part (1) Recall that an edge $xy \in \mathrm{UDG}(V)$ is contained in $\mathrm{PDT}(V)$ if either $xy \in \mathrm{GG}(V) \cap \mathrm{UDG}(V)$, or if there exists a circle $\mathrm{C}(x,z,y)$ which is empty of nodes from $\mathrm{N}_1(x) \setminus \{x,y,z\}$ and where $\sin(\measuredangle xzy) \geq \|xy\|/\mathrm{R}$.

Since uv is an edge in $\mathrm{PuDel}(V)$ it holds that $\overrightarrow{uv}$ and $\overrightarrow{vu}$ are locally detectable direct local Delaunay edges, i.e., there exists a point $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ with $\|up\| = \|vp\| \leq \mathrm{R}/2$. By Lemma 6.21, $p \in \mathrm{VR}_{\mathrm{N}_1(v)}(v) \cap \mathrm{VR}_{\mathrm{N}_1(v)}(u)$. Hence, $\mathrm{CLVE}(u,v)$ is a non-empty set and by Lemma 6.23, it follows in turn that $|\mathrm{CLVE}(u,v)| > 1$. Also note that it cannot be the case that $\mathrm{CLVE}(u,v)$ is equivalent to the straight line $\mathrm{B}(u,v)$, as this would contradict the non-collinearity assumption regarding set V. Consequently, in the case under consideration $\mathrm{CLVE}(u,v)$ can exclusively be of the following two types. It is either

1a) a line segment $[a,b] \in \mathrm{B}(u,v)$, where $a \neq b$, or

1b) it is a half-line lying on $\mathrm{B}(u,v)$.

Case 1a) $\mathrm{CLVE}(u,v)$ is a line segment $[a,b] \subset \mathrm{B}(u,v)$ as illustrated by Figure 6.5. Since $a \neq b$, at least one of the two points has to be different from m. Assume w.l.o.g. that $b \neq m$ and that b is below $\overline{uv}$ (the other case is symmetric can be proven analogously). This assumption implies that $\mathrm{y}(a) > \mathrm{y}(b)$.

If $\mathrm{y}(a) > \mathrm{y}(m)$, then there exists $[a,b] \in \mathrm{CLVE}(u,v)$ with $m \in [a,b]$ and $m \neq a,b$. Lemma 6.24 yields that $uv \in \mathrm{GG}(V) \cap \mathrm{UDG}(V)$, which implies that $uv \in \mathrm{PDT}(V)$.

If $\mathrm{y}(a) \leq \mathrm{y}(m)$, then either $a = m$, or a lies strictly below $\overline{uv}$, but within the area $\mathrm{C}_{\mathrm{R}/2}(u) \cap \mathrm{C}_{\mathrm{R}/2}(v)$, since $uv \in \mathrm{PuDel}(V)$. Moreover, in either case $\|ma\| \leq \|mb\|$ holds.

Observe that point a is a Voronoi vertex w.r.t. either $\mathrm{VD}(\mathrm{N}_1(u))$, or $\mathrm{VD}(\mathrm{N}_1(v))$. Assume the former holds. In accordance with Lemma 6.25, there must exist a node l s.t. a is the point of intersection of $\mathrm{VR}_{\mathrm{N}_1(u)}(u)$, $\mathrm{VR}_{\mathrm{N}_1(u)}(v)$, and $\mathrm{VR}_{\mathrm{N}_1(u)}(l)$.

Node l must be contained by $\mathrm{D}(u,v)$ for the following reason. Firstly, point a is the circumcenter of $\mathrm{C}(u,v,l)$ and while $\mathrm{y}(a) \leq \mathrm{y}(m)$, the arc of $\mathrm{C}(u,v,l)$ which is contained in the upper half-plane w.r.t. $\overline{uv}$ is entirely contained by $\mathrm{D}(u,v)$. Secondly, $\mathrm{y}(a) > \mathrm{y}(b)$ and hence, l must be located in the upper half-plane w.r.t. $\overline{uv}$.

From the above it follows that $l \in (\mathrm{N}_1(u) \cap \mathrm{N}_1(v))$ and hence, l is a generator node for the Voronoi vertex a in $\mathrm{VD}(\mathrm{N}_1(v))$ as well. By Lemma 6.26 it holds that $\mathrm{C}(u,v,l)$ is empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u,v,l\}$. For the remainder let $\alpha = \measuredangle ulv$.

Consider the case where $a = m$ (see Figure 6.5 for an illustration). In this case l must be located on the boundary of $\mathrm{D}(u,v)$. Hence, $\mathrm{D}(u,v)$ and $\mathrm{C}(u,v,l)$ coincide and $\mathrm{D}(u,v)$ must also be empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u,v,l\}$. Moreover, $\alpha = \pi/2$ (by

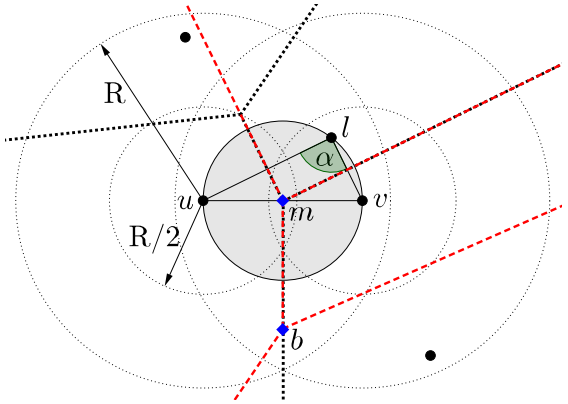


Figure 6.5: Illustration for the proof of Theorem 6.27, Part (1), case $a = m$. CLVE(u, v) is given by $\overline{mb}$. VD$(\mathrm{N}_1(u))$ and VD$(\mathrm{N}_1(v))$ are represented by the dotted and dashed lines, respectively. The shaded area represents D(u, v).

Thales' theorem) and thus, $\sin(\alpha) = 1$. With $\|uv\| \leq \mathrm{R}$, $\sin(\alpha) = 1 \geq \|uv\|/\mathrm{R}$ holds. It can therefore be concluded that $uv \in \mathrm{PDT}(V)$.

In the remaining case, both points a and b are located strictly below $\overline{uv}$ (see Figure 6.6). Because $\overrightarrow{uv}$ is locally detectable and because of the assumption that $\mathrm{y}(a) > \mathrm{y}(b)$, it must hold that $a \in \mathrm{C}_{\mathrm{R}/2}(u)$, whereas b may or may not be contained in $\mathrm{C}_{\mathrm{R}/2}(u)$. The former implies that l must be contained in the interior of D(u, v). Recall that C(u, v, l) with center a is empty of nodes from $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u, v, l\}$. If it can be shown that $\sin(\alpha) \geq \|uv\|/\mathrm{R}$, for all possible positions of a, then $uv \in \mathrm{PDT}(V)$ follows immediately.

Suppose a is moved along CLVE(u, v) towards the lower point of intersection of $\mathrm{C}_{\mathrm{R}/2}(u)$ and $\mathrm{C}_{\mathrm{R}/2}(v)$, denoted by p. Figure 6.6a shows a possible initial position, whereas Figure 6.6b shows the situation after moving a, such that $a = p$.

It can be observed that with monotonically decreasing value $\mathrm{y}(a)$, angle α increases monotonically, whereas $\sin(\alpha)$ decreases monotonically, because $\pi/2 < \alpha < \pi$. Moreover, α reaches its maximum and $\sin(\alpha)$ reaches its minimum, if $a = p$, since only those cases need to be considered where $\overrightarrow{uv}$ remains locally detectable, i.e., cases where $\|ua\| \leq \mathrm{R}/2$. Hence, it is sufficient to consider the case where $a = p$. The application of the *law of sines* yields:

$$\sin(\alpha) = \frac{\|uv\|}{2 \cdot \|ua\|} = \frac{\|uv\|}{2 \cdot \mathrm{R}/2} = \frac{\|uv\|}{\mathrm{R}}.$$

Thus, $\sin(\alpha) \geq \|uv\|/\mathrm{R}$ for all possible positions of a under consideration and it can be concluded that $uv \in \mathrm{PDT}(V)$.

Case 1b) The case where CLVE(u, v) is a half-line is a special case of Case 1a) and can be proven equivalently. Let p denote the unique endpoint of CLVE(u, v) and apply the same line of argumentation as used in the proof of Case 1a) on point p, instead of point a. Then, $uv \in \mathrm{PDT}(V)$ follows for this case as well.

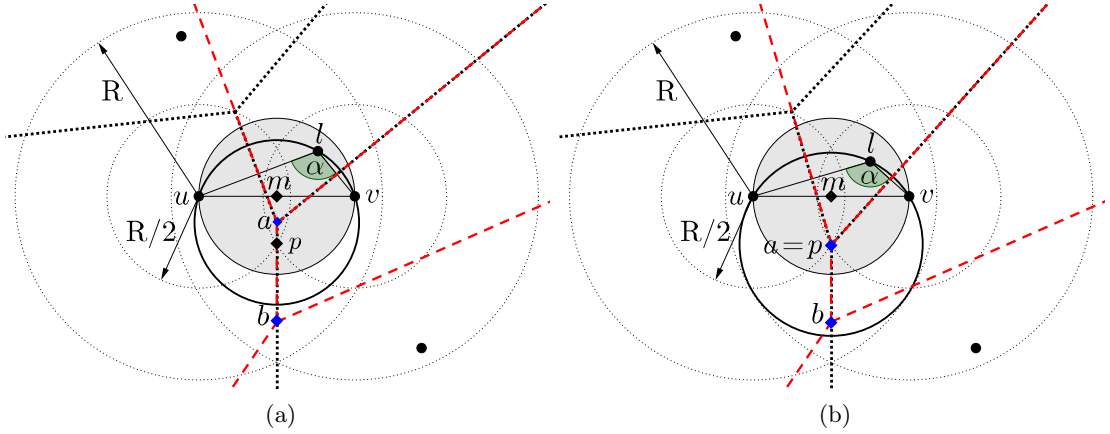


Figure 6.6: Illustration for the proof of Theorem 6.27, Part (1), case $a \neq m$. (a) Case where a is strictly below $\overline{uv}$. (b) Case where $a = p$, where p is the lower point of intersection of $\mathrm{C}_{\mathrm{R}/2}(u)$ and $\mathrm{C}_{\mathrm{R}/2}(v)$. In both figures $\mathrm{CLVE}(u, v)$ is given by $\overline{ab}$. $\mathrm{VD}(\mathrm{N}_1(u))$ and $\mathrm{VD}(\mathrm{N}_1(v))$ are represented by the dotted and dashed lines, respectively. The shaded area represents $\mathrm{D}(u, v)$ and the bold black circle represents $\mathrm{C}(u, v, l)$.

Part (2) For the remainder of this proof let $uv \in \mathrm{PDT}(V)$. Recall that $\overline{uv}$ is assumed to be parallel to the x-axis. Let l be the angle maximizing node w.r.t. $\overline{uv}$, i.e., the node maximizing the angle $\measuredangle ulv$ among all nodes in $(\mathrm{N}_1(u) \cup \mathrm{N}_1(v)) \setminus \{u, v\}$. Note that under the assumption that V is non-collinear, the angle maximizing node is uniquely defined. Let $\alpha = \measuredangle ulv$ and refer to Figure 6.7 for an illustration of the following proof.

According to the definition of PDT, $uv \in \mathrm{UDG}(V)$ and uv is either an edge of the Gabriel graph, or circle $\mathrm{C}(u, v, l)$ is empty of nodes from $\mathrm{N}_1(u) \setminus \{u, v, l\}$ and satisfies $\sin(\alpha) \geq \|uv\|/\mathrm{R}$.

If $uv \in \mathrm{GG}(V) \cap \mathrm{UDG}(V)$, then $uv \in \mathrm{PuDel}(V)$, since $\mathrm{GG}(V) \cap \mathrm{UDG}(V) \subseteq \mathrm{PuDel}(V)$ by Lemma 6.9.

Therefore, assume that $uv \notin \mathrm{GG}(V)$. In order to prove that $uv \in \mathrm{PuDel}(V)$ it suffices to show that $\overline{uv}$ is a locally detectable directed local Delaunay edge. That is, it is required to show:

(a) $\mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v) \neq \emptyset$ (i.e., $\overline{uv}$ is a directed local Delaunay edge), and

(b) $\exists p \in \mathbb{R}^2 : p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ with $\|up\| \leq \mathrm{R}/2$ (the directed local Delaunay edge is locally detectable).

Obviously, (b) implies (a) and it suffices to prove existence of a point p as defined in (b).

In the case under consideration $uv \notin \mathrm{GG}(V)$ and hence, angle maximizing node l is either on the boundary of or in the interior of $\mathrm{D}(u, v)$. Therefore, $\|ul\| < \|uv\|$ which implies that $l \in \mathrm{N}_1(u)$. Let p denote the midpoint of circle $\mathrm{C}(u, v, l)$ and observe that $p \in \mathrm{B}(u, v)$ as shown in Figure 6.7. Because $uv \in \mathrm{PDT}(V)$, the interior of $\mathrm{C}(u, v, l)$ is

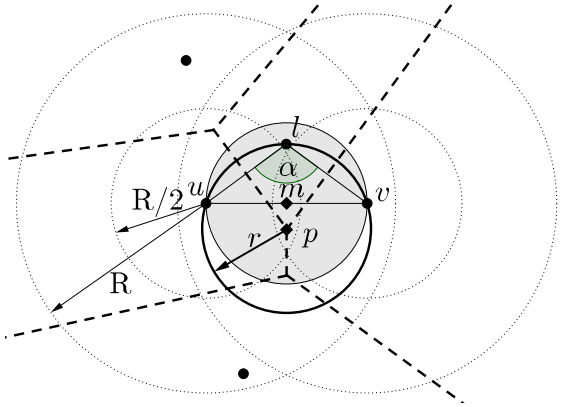


Figure 6.7: Illustration for the proof of Theorem 6.27 Part (2). The dashed lines represent $\mathrm{VD}(\mathrm{N}_1(u))$. The shaded circle is $\mathrm{D}(u,v)$, whereas the bold black circle represents $\mathrm{C}(u,v,l)$.

empty of nodes from $\mathrm{N}_1(u) \setminus \{u,v,l\}$ and $\mathrm{C}(u,v,l)$ is entirely contained by the unit disk centered at u. Thus, $\mathrm{C}(u,v,l)$ is empty of nodes from $V \setminus \{u,v,l\}$. From Lemma 6.3 it follows that p is a Voronoi vertex in $\mathrm{VD}(V)$ joining the Voronoi regions $\mathrm{VR}_V(u)$, $\mathrm{VR}_V(v)$, and $\mathrm{VR}_V(l)$. By Lemma 6.5 $p \in \mathrm{VR}_{\mathrm{N}_1(u)}(u) \cap \mathrm{VR}_{\mathrm{N}_1(u)}(v)$ holds as well. It remains to show that p is a witness for $\overrightarrow{uv}$'s local detectability for all possible positions of l.

If l lies on the boundary of $\mathrm{D}(u,v)$, then $p = m$. Since $\|uv\| \leq \mathrm{R}$, $\|up\| \leq \mathrm{R}/2$ holds. Therefore, assume l lies strictly inside $\mathrm{D}(u,v)$ and assume in addition w.l.o.g. that l is contained in the upper half-circle of $\mathrm{D}(u,v)$ w.r.t. $\overline{uv}$. Because $uv \in \mathrm{PDT}(V)$, $\sin(\alpha) \geq \|uv\|/\mathrm{R}$ holds. Observe that with increasing value of α, which is the case if the distance between l and the midpoint m of $\overline{uv}$ decreases, the difference between $\sin(\alpha)$ and $\|uv\|/\mathrm{R}$ decreases. Moreover, with increasing angle α, the radius of $\mathrm{C}(u,v,l)$ is monotonically increasing and so is the distance between u and p. While uv is an edge in $\mathrm{PDT}(V)$, $\|up\|$ is maximized if $\sin(\alpha) = \|uv\|/\mathrm{R}$ holds. Therefore, it suffices to show that $\|up\| \leq \mathrm{R}/2$ holds in the latter case. Let r denote the radius of $\mathrm{C}(u,v,l)$. The law of sines yields:

$$\mathrm{R} = \frac{\|uv\|}{\sin(\alpha)} = 2 \cdot r = 2 \cdot \|up\|.$$

That is, independent of l's position, while $uv \in \mathrm{PDT}(V)$, p is a witness for the local detectability of the directed local Delaunay edge $\overrightarrow{uv}$ and hence, $uv \in \mathrm{PuDel}(V)$ holds. □

6.1.4 Discussion and Implications

The equivalence of PuDel and PDT has various implications, which are discussed subsequently. First and foremost, the equivalence of PuDel and PDT implies the following important corollary.

Corollary 6.28. *Let V be a set of nodes such that $\mathrm{UDG}(V)$ is a connected graph. Then, $\mathrm{PDT}(V)$ is a Euclidean $(\frac{1+\sqrt{5}}{4}\pi^2)$-spanner of $\mathrm{UDG}(V)$.*

Proof. By Theorem 6.20, PuDel(V) is a Euclidean ($\frac{1+\sqrt{5}}{4}\pi^2$)-spanner of UDG($V$). By Theorem 6.27, PuDel(V) = PDT(V). Hence, PDT(V) is a Euclidean ($\frac{1+\sqrt{5}}{4}\pi^2$)-spanner of UDG($V$). □

This result answers the formerly open question, whether PDT is or is not a constant stretch Euclidean spanner for UDG. Previously, some authors [59, 62–65] stated this to be unknown, whereas others [66, 67] even claimed that PDT is no constant stretch spanner at all. In addition this results implies a constant Euclidean spanning ratio for PDT's 2-hop counterpart (see Definition 3.2).

Theorem 6.29. $\mathrm{PDT}_2(V)$ *is a Euclidean* ($\frac{1+\sqrt{5}}{4}\pi^2$)*-spanner of* UDG($V$).

Proof. It holds that $\mathrm{PDT}(V) \subseteq \mathrm{PDT}_2(V)$ [113]. Since PDT(V) is a Euclidean ($\frac{1+\sqrt{5}}{4}\pi^2$)-spanner of UDG($V$) by Theorem 6.28, this also holds for $\mathrm{PDT}_2(V)$. □

There are several applications of PDT which may benefit from the theoretical guarantees provided in this section. For example, Tan [56] uses PDT as a component for a distributed autonomous deployment algorithm for mobile robots in mobile sensor networks. Deng and Stojmenović [53] as well as Li. et al [55] propose to use PDT for application in geographic hashing. It will be the subject of future work to investigate in how far the constant spanning properties of PDT extend and can improve these results.

Finally, the fact that PDT is a spanner also has direct and important impacts on recent results in beaconless Recovery routing. These are discussed and presented in detail in Chapter 7.

6.2 Reactive Local Construction of the Partial Delaunay Triangulation

In the following it is shown that a node's local view on PDT can be constructed by an $\mathcal{O}$-reactive local view topology control algorithm, called *reactivePDT* (rPDT).

Algorithm rPDT extends and improves the timer-based contention principles of BFP [18, 107] in the following sense. Like BFP, algorithm rPDT is also an RTS-CTS-based protocol that uses distance-based delay timers and requires no additional input other than the geographic position of its executing node as well as the unit transmission radius. However, unlike the select-and-protest-based approaches BFP and GDBF, the algorithm presented here does not require an additional protest phase and consists solely of a selection phase. This is achieved by making use of re-adjustable delay timers whose timeouts are modified online, during algorithm execution.

The main contribution—the description and correctness proof of algorithm rPDT—is presented in Section 6.2.2. Beforehand, in Section 6.2.1 important facts and definitions are established. Lastly, Section 6.2.3 discusses algorithmic properties and implications to the state of research.

6.2.1 Preliminaries

As in the previous section, let $V \subset \mathbb{R}^2$ be a distinct and finite set of nodes, such that no four nodes in V are cocircular and no three nodes in V are collinear. $\mathrm{UDG}(V)$ denotes the unit disk graph over V. For simplicity of the proofs and w.l.o.g. it is henceforth assumed that the unit disk radius is equal to one, i.e., $\mathrm{R} = 1$.

Elementary Geometric Constructions and Proofs

Given three non-collinear points $u, v, w \in \mathbb{R}^2$, let $\mathrm{H}^w(u, v)$ denote the *open half-plane* which contains w and that is bounded by the straight line $\ell(u, v)$. For simplicity, the following notation is used frequently. Let $A \subset \mathbb{R}^2$ be some area, e.g., a half-plane or a circle. Then $\overline{A}$ refers to the *complementary area* of A, given by $\mathbb{R}^2$ without point set A.

A node $w \in \mathrm{D}(u, v)$ is referred to as *angle maximizing node w.r.t. uv* (or simply *angle maximizing node*), if and only if $\measuredangle uwv \geq \measuredangle uzv$, for all $z \in \mathrm{D}(u, v)$, $z \neq u, v$.

The proofs of Lemma 6.30 & 6.31 are similar to the proof of Lemma 1 in [50] and are given here only for the sake of completeness.

Lemma 6.30. *Let $u, v, w \in V$ s.t. $w \in \mathrm{D}(u, v)$ and let $\alpha = \measuredangle uwv$, then*

$$\frac{\|uv\|}{\sin(\alpha)} = \mathrm{d}_{uvw}\,.$$

Proof. Because $w \in \mathrm{D}(u, v)$, it holds that $\frac{\pi}{2} \leq \alpha \leq \pi$ and hence, it also holds that $\sin(\alpha) = \cos(\alpha - \frac{\pi}{2})$. Let z be the point on $\mathrm{C}(u, v, w)$, s.t. $\overline{uz} = \mathrm{d}_{uvw}$ and let $\beta = \measuredangle zuv$. Then, because of symmetry of angles $\beta = \alpha - \frac{\pi}{2}$ holds and hence,

$$\sin(\alpha) = \cos(\alpha - \frac{\pi}{2}) = \cos(\beta) = \frac{\|uv\|}{\mathrm{d}_{uvw}}\,.$$

□

Lemma 6.31. *Let $u, v, w \in V$ s.t. $w \in \mathrm{D}(u, v)$ and let $\alpha = \measuredangle uwv$, then $\sin(\alpha) \geq \|uv\|$ if and only if $\mathrm{d}_{uvw} \leq 1$.*

Proof. This follows immediately from Lemma 6.30. □

Lemma 6.32. *Let $w \in \mathrm{D}(u, v)$ be the angle maximizing node w.r.t. uv and let $\hat{w} \in \mathrm{D}(u, v)$ be another node. It holds that $\mathrm{d}_{uvw} \geq \mathrm{d}_{uv\hat{w}}$.*

Proof. Let $\alpha = \measuredangle uwv$ and $\hat{\alpha} = \measuredangle u\hat{w}v$. Because w is assumed to be angle maximizing, $\alpha \geq \hat{\alpha}$. This is equivalent to $\sin(\alpha) \leq \sin(\hat{\alpha})$ (note that $\frac{\pi}{2} \leq \alpha, \hat{\alpha} \leq \pi$). With Lemma 6.30 it holds that

$$\mathrm{d}_{uv\hat{w}} = \frac{\|uv\|}{\sin(\hat{\alpha})} \leq \frac{\|uv\|}{\sin(\alpha)} = \mathrm{d}_{uvw}\,.$$

□

Proximity graphs and their equivalence

Next, two alternative definitions of PDT are given. These simplify the proofs given in this section. Definition 6.33 defines PDT using half-planes. In Definition 6.34, this definition is then further modified by replacing the concept of the angle maximizing node with a statement that holds for all nodes contained in a Gabriel circle.

Definition 6.33 (Partial Delaunay triangulation (PDT))**.** Let $uv \in \mathrm{UDG}(V)$ and $w \in V$ the angle maximizing node w.r.t. uv. Edge uv is contained by the *partial Delaunay triangulation* $\mathrm{PDT}(V)$, if and only if either $\mathrm{D}(u,v)$ does not contain any node from $V \setminus \{u,v\}$, or if

1. $\nexists\, x \in V \setminus \{u,v,w\}$ s.t. $x \in \mathrm{C}(u,v,w) \cap \overline{\mathrm{H}^w(u,v)}$, and
2. $\sin(\alpha) \geq \|uv\|$, where $\alpha = \measuredangle uwv$.

Definition 6.34 (Generalized partial Delaunay triangulation (GPDT))**.** An edge $uv \in \mathrm{UDG}(V)$ is contained by the *generalized partial Delaunay triangulation* $\mathrm{GPDT}(V)$, if and only if for all nodes $w \in \mathrm{D}(u,v)$, $w \neq u,v$, it holds that

1. $\nexists\, x \in V \setminus \{u,v,w\}$ s.t. $x \in \mathrm{C}(u,v,w) \cap \overline{\mathrm{H}^w(u,v)}$, and
2. $\sin(\alpha) \geq \|uv\|$, where $\alpha = \measuredangle uwv$.

Let $uv \in \mathrm{UDG}(V)$ s.t. there exist nodes $w \in \mathrm{D}(u,v)$ and $x \in \mathrm{C}(u,v,w) \cap \overline{\mathrm{H}^w(u,v)}$. Edge uv is said to *violate* the first PDT (GPDT) criterion with *angular node* w and *witness node* x. Likewise, edge uv is said to violate the second PDT (GPDT) criterion with *angular node* w.

Next it is shown that PDT and GPDT are in fact equivalent definitions.

Theorem 6.35. *Applied on any* $\mathrm{UDG}(V)$*, the definitions of* PDT *and* GPDT *yield equivalent subgraphs, i.e., for* $u,v \in V$ *it holds that* $uv \in \mathrm{GPDT}(V) \Leftrightarrow uv \in \mathrm{PDT}(V)$.

Proof. "⇒": Consider any edge $uv \in \mathrm{GPDT}(V)$. If $\mathrm{D}(u,v) \cap V \setminus \{u,v\} = \emptyset$, then $uv \in \mathrm{PDT}(V)$. Otherwise, the fact that the two GPDT criteria hold for all nodes contained in $\mathrm{D}(u,v)$ implies that they hold in particular for the angle maximizing node $w \in \mathrm{D}(u,v)$ and thus, $uv \in \mathrm{PDT}(V)$.

"⇐": The remaining case is proven by showing that $uv \notin \mathrm{GPDT}(V) \Rightarrow uv \notin \mathrm{PDT}(V)$.

Let $uv \in \mathrm{UDG}(V)$ be any edge which is not contained in $\mathrm{GPDT}(V)$. Then for at least one node $\hat{w} \in \mathrm{D}(u,v)$, at least one GPDT criterion is violated. Let $w \in \mathrm{D}(u,v)$ be the angle maximizing node w.r.t. uv.

If the first GPDT criterion is violated, then there exists at least one node $x \in \mathrm{C}(u,v,\hat{w})$ that lies on the opposite side of $\hat{w}$ w.r.t. uv. Angle maximizing node w and node $\hat{w}$ are either on the same or on opposite sides w.r.t. uv. Assume the former holds. Then, according to Lemma 6.32, $\mathrm{d}_{uvw} \geq \mathrm{d}_{uv\hat{w}}$, because w is assumed to be angle maximizing. Because w and $\hat{w}$ are on the same side w.r.t. uv, $x \in \mathrm{C}(u,v,\hat{w})$ implies that $x \in \mathrm{C}(u,v,w)$. Thus, the first PDT criterion is violated and hence, $uv \notin \mathrm{PDT}(V)$.

Now assume that w and $\hat{w}$ are on opposite sides w.r.t. uv. Then, because $w, \hat{w} \in \mathrm{D}(u,v)$, it holds that $w \in \mathrm{C}(u,v,\hat{w})$ and $\hat{w} \in \mathrm{C}(u,v,w)$. Again, the first PDT criterion is violated and $uv \notin \mathrm{PDT}(V)$ holds.

Finally, consider the case where the second GPDT criterion is violated. Let $\hat{\alpha} := \measuredangle u\hat{w}v$ and $\alpha := \measuredangle uwv$. By assumption, $\sin(\hat{\alpha}) < \|uv\|$ and hence,

$$\begin{aligned} & 1 \overset{\text{Lemma 6.31}}{<} \mathrm{d}_{uv\hat{w}} \overset{\text{Lemma 6.32}}{\leq} \mathrm{d}_{uvw} \overset{\text{Lemma 6.30}}{=} \frac{\|uv\|}{\sin(\alpha)} \\ \Leftrightarrow \quad & \sin(\alpha) < \|uv\| \,. \end{aligned}$$

Thus, edge uv violates the second PDT criterion and $uv \notin \mathrm{PDT}(V)$. □

6.2.2 Reactive Construction of PDT

In the following a beaconless algorithm for construction of a node's adjacency in PDT is introduced and proven to be correct. When executed by any node u from a given unit disk graph $\mathrm{UDG}(V)$, the algorithm computes $\mathrm{N}_1^{\mathrm{PDT}(V)}(u)$, the one-hop neighborhood of u in $\mathrm{PDT}(V)$. At the time of algorithm execution no node in V is aware of its network neighborhood in $\mathrm{UDG}(V)$. At first, the algorithm is described in detail and thereafter it is proven to be correct.

Algorithm reactivePDT (rPDT)

Any node $u \in V$ is assumed to have a delay timer $\mathrm{t}(u)$ with the following properties: $\mathrm{timeout}_{\mathrm{t}(u)}$ describes the point in time of the timer's expiration, relative to its starting time. That is, if timer $\mathrm{t}(u)$ is started at time t with $\mathrm{timeout}_{\mathrm{t}(u)} = \mathrm{t}_{\max}$, then this timer expires at time $t + \mathrm{t}_{\max}$. Moreover, the delay timer is assumed to be *re-adjustable*, i.e., while it is running, it may be extended (possibly multiple times) by some additional time slice.[2]

Description of execution of algorithm rPDT initiated by $u \in \mathrm{UDG}(V)$:

Node u initializes the output set $\mathrm{N}_1^{\mathrm{PDT}(V)}(u)$ to be the empty set, locally broadcasts an RTS including its geographic position, and schedules termination after expiration of the delay timer $\mathrm{t}(u)$ with $\mathrm{timeout}_{\mathrm{t}(u)} = \mathrm{t}_{\max}$.

Upon reception of an RTS sent by node u, any node $v \in \mathrm{N}_1(u)$ initializes its *set of known nodes* $\mathrm{S}(v)$ to be the empty set and initializes its *current maximal angle* $\alpha_{\max}(v) = \pi/2$. In addition it schedules sending of a CTS response, including its geographic position, upon expiry of delay timer $\mathrm{t}(v)$ with $\mathrm{timeout}_{\mathrm{t}(v)}$ proportional to the diameter of the Gabriel circle $\mathrm{D}(u,v)$, i.e., it sets $\mathrm{timeout}_{\mathrm{t}(v)} = \|uv\| \cdot \mathrm{t}_{\max}$.

If a node $v \neq u$ overhears a CTS sent by node z, then v adds z to its list of known neighbors $\mathrm{S}(v)$. Furthermore, it checks if uv violates any of the two GPDT criteria w.r.t.

[2] For example, if node u sets up the delay timer $\mathrm{t}(u)$ with $\mathrm{timeout}_{\mathrm{t}(u)} = 1/2 \cdot \mathrm{t}_{\max}$ at time t, then as long as the timer is still running, node u may extend the delay time dynamically, say by $1/4 \cdot \mathrm{t}_{\max}$. Then this timer expires at time $t + 3/4 \cdot \mathrm{t}_{\max}$.

the set of known nodes $\mathrm{S}(v)$. If a violation is detected, then v cancels its timer $\mathrm{t}(v)$ and refrains from sending its CTS. Otherwise, if z is contained in $\mathrm{D}(u,v)$ and satisfies $\alpha = \measuredangle uzv > \alpha_{\max}(v)$, then node v updates the current maximal angle and re-adjusts the timeout of timer $\mathrm{t}(v)$ proportional to the diameter of the circle $\mathrm{C}(u,z,v)$, i.e., sets $\mathrm{timeout}_{\mathrm{t}(v)} = \frac{\|uv\|}{\sin(\alpha)} \cdot \mathrm{t}_{\max}$. The rationale behind this is that uv is now known not to be an edge of the Gabriel graph, but it may still be contained in $\mathrm{PDT}(V)$.

Whenever node u receives a CTS sent by a node v, then node u adds v to its output set $\mathrm{N}_1^{\mathrm{PDT}(V)}(u)$. Once its delay timer is timed-out, it terminates.

A detailed pseudocode description of algorithm rPDT is given in Algorithm 1.

Algorithm 1 reactivePDT (rPDT)

Variables: u is the initiating node, R is the unit disk transmission radius, and $\mathrm{t}_{\max} > 0$ is a constant known to all nodes.

Definition of message types

RTS(u) request for construction of $\mathrm{N}_1^{\mathrm{PDT}(V)}(u)$ by node u, represented by its position.

CTS(v) response from a PDT-neighbor v of u, represented by its position.

Action of initiator u

1: $\mathrm{N}_1^{\mathrm{PDT}(V)}(u) \leftarrow \emptyset$ ▷ Initialize output set
2: Locally broadcast RTS(u)
3: Schedule termination after expiration of delay timer $\mathrm{t}(u)$ with $\mathrm{timeout}_{\mathrm{t}(u)} \leftarrow \mathrm{t}_{\max}$ and process incoming messages in the meantime
4: **on** reception of CTS(z) **do**
5: $\quad \mathrm{N}_1^{\mathrm{PDT}(V)}(u) \leftarrow \mathrm{N}_1^{\mathrm{PDT}(V)}(u) \cup \{z\}$

Action of neighbor $v \neq u$

6: **on** reception of RTS(u) **do**
7: $\quad \mathrm{S}(v) \leftarrow \emptyset$ ▷ Initialize *set of known nodes*
8: $\quad \alpha_{\max}(v) \leftarrow \pi/2$ ▷ Initialize *current maximal angle*
9: $\quad$ Schedule local broadcast of CTS(v) upon expiry of delay timer $\mathrm{t}(v)$ with $\mathrm{timeout}_{\mathrm{t}(v)} \leftarrow \|uv\| \cdot \mathrm{t}_{\max}$, and process incoming messages in the meantime
10: **on** reception of CTS(z) **do**
11: $\quad \mathrm{S}(v) \leftarrow \mathrm{S}(v) \cup \{z\}$ ▷ Update *set of known neighbors*
12: $\quad$ Check if edge uv violates any of the two GPDT criteria w.r.t. $\mathrm{S}(v)$
13: $\quad$ **if** a violation is detected **then**
14: $\quad\quad$ Cancel delay timer $\mathrm{t}(v)$ and cancel local broadcast of CTS(v)
15: $\quad$ **else**
16: $\quad\quad \alpha \leftarrow \measuredangle uzv$
17: $\quad\quad$ **if** $\alpha > \alpha_{\max}(v)$ **then** ▷ Node z maximizes the angle w.r.t. uv
18: $\quad\quad\quad \alpha_{\max}(v) \leftarrow \alpha$
19: $\quad\quad\quad$ Re-adjust the timeout of timer $\mathrm{t}(v)$ proportional to the diameter of circle $\mathrm{C}(u,z,v)$ by setting $\mathrm{timeout}_{\mathrm{t}(v)} \leftarrow \frac{\|uv\|}{\sin(\alpha)} \cdot \mathrm{t}_{\max}$

Structure of the Correctness Proof

Prior to proving reactivePDT to be correct, the most important ideas, observations, and the proof's structure are outlined.

The key observation is that once the executing node u has broadcasted its RTS, u's neighbors reply with a CTS message if and only if they are PDT-neighbors of u (Theorem 6.47). Proving that each PDT-neighbor actually replies is rather simple; however, proving that non-PDT-neighbors do not reply, requires a deeper analysis of possible worst-case situations.

Recall that initially none of the nodes is aware of its neighborhood. Thus, a non-PDT-neighbor v of u does not reply with a CTS message, if and only if it overhears one or more CTS messages prior to expiration of its timer, which give evidence of v not being a PDT-neighbor of u. Nodes not sending a CTS are henceforth called *hidden nodes*, for they remain invisible to all other nodes. Node v becomes a hidden node if it either overhears a message of *angular node* w, s.t. u, v, w violate the second GPDT criterion, or if it overhears messages of *angular node* w and *witness node* x, s.t. u, v, w, x violate the first or both GPDT criteria. Therefore, such nodes' existence as well as their timers' early expirations have to be proven. Note that both the angular and the witness node have to be PDT-nodes (i.e., non-hidden nodes) in order to actually send a message.

The problem of showing the existence of such PDT-neighbors results from the following worst-case situation of a cascaded sequence of hidden nodes (see Figure 6.9b for an illustration). Assume uv is not a GPDT edge. The angle maximizing node w_i w.r.t. uv may itself be a hidden node. The angle maximizing node w_{i+1} w.r.t. uw_i may also be a hidden node, and so on.

Such situations and the existence of a suitable angular node for uv are considered in Lemmata 6.39– 6.42. The existence of a suitable witness node is proven in Lemmata 6.43–6.45. Lemma 6.46 then deals with timers' early expirations. In Lemma 6.36 and 6.37, geometric properties are proven that are used frequently throughout the other proofs.

Correctness Proof

Lemma 6.36. *Let $u, v, w \in V$ s.t. $w \in \mathrm{D}(u, v)$ is the angle maximizing node w.r.t. uv. The interior of $\mathrm{C}(u, v, w) \cap \mathrm{H}^w(u, v)$ is empty of nodes from V.*

Proof. For the sake of contradiction assume the existence of $z \in \mathrm{C}(u, v, w) \cap \mathrm{H}^w(u, v)$. Then $\measuredangle uwv < \measuredangle uzv$ and w is not angle maximizing, contradicting the assumption. □

Lemma 6.37. *Let $u, v, w, y \in V$ s.t. $w \in \mathrm{D}(u, v)$ and $y \in \mathrm{D}(u, w)$ are the angle maximizing nodes w.r.t. uv and uw, respectively. It holds that*

(i) $\mathrm{d}_{uyv} \geq \mathrm{d}_{uyw}$, *and*

(ii) $\mathrm{C}(u, y, w) \cap \mathrm{H}^w(u, y) \subseteq \mathrm{C}(u, y, v) \cap \mathrm{H}^w(u, y)$.

Proof. Assume w.l.o.g. that edge uw is parallel to the y-axis and that $\mathrm{y}(w) > \mathrm{y}(u)$ (see Figure 6.8 for an illustration). Let $\mathcal{L}$ denote the straight line which is orthogonal to edge

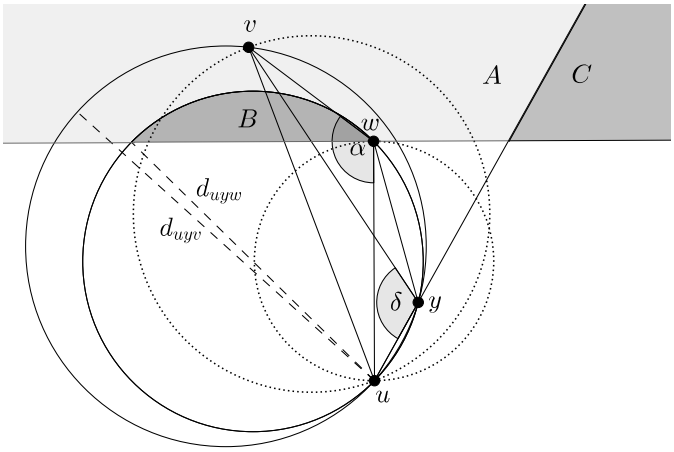


Figure 6.8: Illustration of areas and node locations used in the proof of Lemma 6.37. Gabriel circles are dotted and diamaters are given by dashed lines.

uw and that passes through w. Define the following areas: Let A be the closed half-plane bounded by $\mathcal{L}$ and which does not contain node u, let $B := A \cap \mathrm{D}^{\circ}(u, y, w)$, and let $C := A \cap \overline{\mathrm{H}^{w}(u, y)}$. Moreover, let $\alpha = \measuredangle uwv$ and $\delta = \measuredangle uyv$.

With $w \in \mathrm{D}(u, v)$, $\alpha \geq \pi/2$ holds and thus, $v \in A$. Because w is assumed to be angle maximizing w.r.t. uv it also holds that $\alpha \geq \delta$.

Assume $v \in B$. Let $\hat{v}$ be the point of intersection of the straight line $\ell(v, y)$ with the boundary of $\mathrm{C}(u, y, w)$ in area A and denote the angle $\measuredangle uw\hat{v}$ by $\hat{\alpha}$. Because u, y, w, and $\hat{v}$ are located on the boundary of $\mathrm{C}(u, y, w)$ and $\hat{v} \in \ell(v, y)$ it holds that $\hat{\alpha} = \delta$. In addition, $v \in B$ and $\hat{v} \notin B$ implies $\|v\hat{v}\| > 0$. Then $\alpha < \hat{\alpha} = \delta$ holds, which is a contradiction to w being angle maximizing w.r.t. uv. Thus, $v \in A \setminus B$ must hold from which claim (i) follows directly.

Now, assume that $v \in C$. Then y is contained in the triangle defined by u, v, w (because of the definition of area C), which implies that $\delta > \alpha$. But this contradicts again the assumption that w is angle maximizing w.r.t. uv and thus, $v \notin C$. With $v \notin B \cup C$, claim (ii) follows immediately. □

Definition 6.38 (Hidden node sequence)**.** Let uv be any edge in the unit disk graph $\mathrm{UDG}(V)$. Denote by $\mathrm{HNS}(u, v) := \langle v = w_0, w_1, ..., w_k \rangle$, $k \geq 0$, a non-extendable sequence of nodes from V, such that

- $uw_i \notin \mathrm{GPDT}(V)$, for $0 \leq i \leq k$, and
- $w_{i+1} \in \mathrm{D}(u, w_i)$ is angle maximizing w.r.t. uw_i, for $0 \leq i < k$.

Such a sequence is called a *hidden node sequence* since these nodes do not send a CTS during algorithm execution and hence, remain hidden for all other nodes, in particular for v.

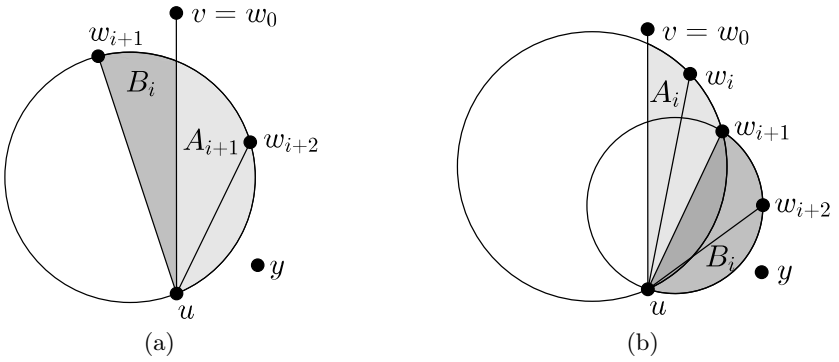


Figure 6.9: Illustrations for the proofs of Lemma 6.40. (a) Representation of Case (i) where $w_{i+1} \in \overline{\mathrm{H}^y(u,v)}$. Area A_{i+1} is part of area B_i. (b) Representation of Case (ii) where $w_{i+1} \in \overline{\mathrm{H}^y(u,v)}$.

Lemma 6.39. *Let $uv \in \mathrm{UDG}(V)$ be an edge with $\mathrm{HNS}(u,v) = \langle v = w_0, w_1, ..., w_k = w \rangle$, $k \geq 0$. Then, there exists an angle maximizing node $y \in \mathrm{D}(u,w)$ w.r.t. uw, such that $uy \in \mathrm{GPDT}(V)$.*

Proof. Because $uw \notin \mathrm{GPDT}(V)$, there exists an angle maximizing node $y \in \mathrm{D}(u,w)$. If $uy \notin \mathrm{GPDT}(V)$, then $\langle w_0, w_1, ..., w_k, y \rangle$ is a valid hidden node sequence. But this contradicts that a hidden node sequence is non-extendable and hence, the claim holds. □

Lemma 6.40. *Let $uv \in \mathrm{UDG}(V)$ and let $\langle v = w_0, w_1, ..., w_k = w \rangle$, where $k \geq 0$, be a sequence of nodes s.t. $w_{i+1} \in \mathrm{D}(u, w_i)$, $0 \leq i < k$, is angle maximizing w.r.t. uw_i. Let $y \in \mathrm{D}(u,w)$ be the angle maximizing node w.r.t. uw. Then $\mathrm{D}^{\mathrm{o}}(u,y,w) \cap \mathrm{H}^y(u,v)$ is empty of nodes from V.*

Proof. For simplicity, let $w_{k+1} := y$. Define area A_i as follows:

$$A_i := \mathrm{D}^{\mathrm{o}}(u, w_{i+1}, w_i) \cap \mathrm{H}^y(u,v) \cap \mathrm{H}^{w_{i+1}}(u,v)\,.$$

Proof by induction on index i.
I.h.: For an arbitrary but fixed i it holds that area A_i is empty of nodes from V.
B.c. ($i = 0$): Area A_0 is defined as $A_0 = \mathrm{D}^{\mathrm{o}}(u, w_1, w_0) \cap \mathrm{H}^y(u,v) \cap \mathrm{H}^{w_1}(u,v)$. Node w_1 is either contained in $\overline{\mathrm{H}^y(u,v)}$, or in $\mathrm{H}^y(u,v)$.

Assume $w_1 \in \overline{\mathrm{H}^y(u,v)}$ holds. Then $\mathrm{H}^y(u,v) \cap \mathrm{H}^{w_1}(u,v) = \emptyset$. Thus, A_0 is empty of nodes from V.

Now, assume $w_1 \in \mathrm{H}^y(u,v)$ holds. Then, w_1 and y lie within the same half-plane w.r.t. uv and it holds that $\mathrm{H}^y(u,v) = \mathrm{H}^{w_1}(u,v)$. Then, $A_0 = \mathrm{D}^{\mathrm{o}}(u, w_1, w_0 = v) \cap \mathrm{H}^{w_1}(u,v)$. Because $w_1 \in \mathrm{D}(u,v)$ is assumed to be angle maximizing w.r.t. uv, Lemma 6.36 implies that A_0 is empty of nodes from V.
I.s. ($i \to i+1$): Node w_{i+2} is either contained in $\overline{\mathrm{H}^y(u,v)}$, or in $\mathrm{H}^y(u,v)$.

If $w_{i+2} \in \overline{\mathrm{H}^y(u,v)}$, then $\mathrm{H}^y(u,v) \cap \mathrm{H}^{w_{i+2}}(u,v) = \emptyset$ and A_{i+1} is empty of nodes. Otherwise, $w_{i+2} \in \mathrm{H}^y(u,v)$ holds. Two cases may be distinguished, namely,

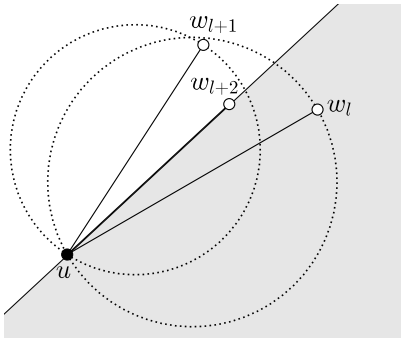


Figure 6.10: Illustration for the proof of Lemma 6.41. Node w_l is contained in $\overline{\mathrm{H}^{w_{l+1}}(u, w_{l+2})}$, which is given by the shaded area. The dotted circles represent $\mathrm{D}(u, w_l)$ and $\mathrm{D}(u, w_{l+1})$, respectively. Hidden nodes are white.

(i) $w_{i+1} \in \overline{\mathrm{H}^y(u, v)}$ (see Figure 6.9a), and

(ii) $w_{i+1} \in \mathrm{H}^y(u, v)$ (see Figure 6.9b).

Case (i): Because $w_{i+2} \in \mathrm{D}(u, w_{i+1})$ is angle maximizing w.r.t. uw_{i+1}, Lemma 6.36 implies that area $B_i := \mathrm{D}^{\mathrm{o}}(u, w_{i+2}, w_{i+1}) \cap \mathrm{H}^{w_{i+2}}(u, w_{i+1})$ is empty of nodes from V. Moreover, because $w_{i+2} \in \mathrm{H}^y(u, v)$ and $w_{i+1} \in \overline{\mathrm{H}^y(u, v)}$ holds, it follows that $A_{i+1} \subseteq B_i$ and thus, A_{i+1} is empty of nodes from V.

Case (ii): By i.h., area A_i is empty of nodes from V. In addition, because w_{i+1} and y lie within the same half-plane w.r.t. uv it holds that $\mathrm{H}^y(u, v) = \mathrm{H}^{w_{i+1}}(u, v)$. Then, $A_i = \mathrm{D}^{\mathrm{o}}(u, w_{i+1}, w_i) \cap \mathrm{H}^{w_{i+1}}(u, v)$. The fact that A_i is empty of nodes from V implies in particular that $w_{i+2} \notin A_i$. Hence, $\mathrm{C}(u, w_{i+2}, w_{i+1}) \cap \overline{\mathrm{H}^{w_{i+2}}(u, w_{i+1})} \cap \mathrm{H}^y(u, v) \subseteq A_i$ (see Figure 6.9b). Moreover, by Lemma 6.36 it holds that area $B_i := \mathrm{D}^{\mathrm{o}}(u, w_{i+2}, w_{i+1}) \cap \mathrm{H}^{w_{i+2}}(u, w_{i+1})$ is empty of nodes from V. From the two latter observations it can be concluded that area $A_{i+1} \subseteq A_i \cup B_i$ is also empty of nodes.

Termination ($i = k$): The i.h. implies that area $A_k = \mathrm{D}^{\mathrm{o}}(u, w_{k+1}, w_k) \cap \mathrm{H}^y(u, v) \cap \mathrm{H}^{w_{k+1}}(u, v)$ is empty of nodes from V. With $w_{k+1} = y$ and $w_k = w$ it follows that $\mathrm{D}^{\mathrm{o}}(u, y, w) \cap \mathrm{H}^y(u, v)$ is empty of nodes from V. □

Lemma 6.41. *Let $uv \in \mathrm{UDG}(V)$ with $\mathrm{HNS}(u, v) = \langle v = w_0, w_1, ..., w_k = w \rangle$, $k \geq 0$. Let $y \in \mathrm{D}(u, w)$ be the angle maximizing node w.r.t. uw. For all $0 \leq i \leq k$ it holds that $w_i \in \mathrm{H}^w(u, y)$.*

Proof. At first, notice the existence of such a node $y \in \mathrm{D}(u, w)$ according to Lemma 6.39. It is now shown that the assumption that there is at least one node in $\mathrm{HNS}(u, v)$ which is not contained in $\mathrm{H}^w(u, y)$ leads to a contradiction and hence, this cannot be the case.

It trivially holds that $w_k = w$ is located in $\mathrm{H}^{w_k}(u, y)$. Assume there is at least one node in $\langle w_0, w_1, ..., w_{k-1} \rangle$ which is not contained in $\mathrm{H}^{w_k}(u, y)$ and let $0 \leq l \leq k - 1$ be the largest index for which this holds. For convenience, rename y by w_{k+1}.

Assume $w_l \in \overline{\mathrm{H}^{w_{l+1}}(u, w_{l+2})}$ as illustrated by Figure 6.10. By definition of hidden node sequences it holds that $w_{l+1} \in \mathrm{D}(u, w_l)$ and $w_{l+2} \in \mathrm{D}(u, w_{l+1})$. The latter combined with

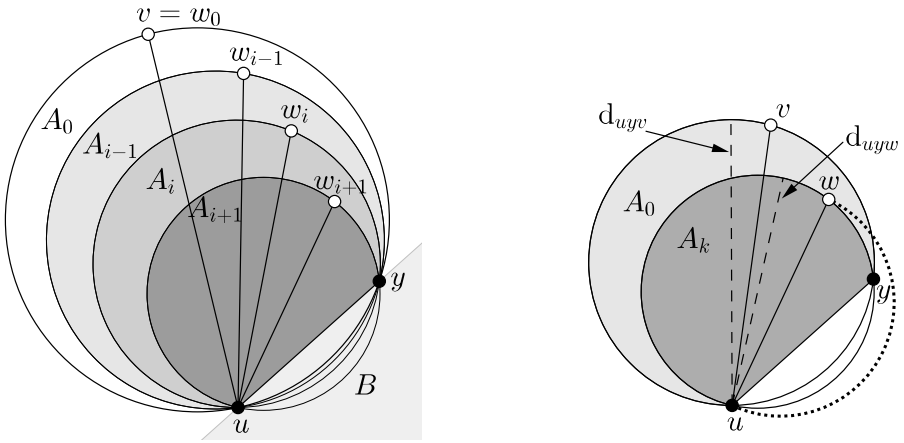


Figure 6.11: Illustrations for the proof of Lemma 6.42. (a) I.s.: $A_i \supseteq A_{i+1}$. (b) Diameter of $\mathrm{C}(u, y, w)$ is at most as large as the diameter of $\mathrm{C}(u, y, v)$. The dotted half-circle represents a half-circle of $\mathrm{D}(u, w)$, diameters are given by dashed lines, and white nodes are hidden nodes.

the assumption that w_l and w_{l+1} are separated by $\ell(u, w_{l+2})$ yields that $w_{l+2} \in \mathrm{D}(u, w_l)$ as well as that $\measuredangle uw_{l+2}w_l > \measuredangle uw_{l+1}w_l$. But this contradicts the assumption that w_{l+1} is the angle maximizing node w.r.t. uw_l and hence, $w_l \in \mathrm{H}^{w_{l+1}}(u, w_{l+2})$ must hold. By choice of w_l it holds true for all $\langle w_{l+1}, ..., w_k \rangle$ that they are contained in $\mathrm{H}^{w_k}(u, y)$. Using the fact that $w_{i+1} \in \mathrm{D}(u, w_i)$, for all $0 \leq i \leq k$, it then holds that $w_l \in \mathrm{H}^{w_k}(u, y)$ which contradicts the assumption. □

Lemma 6.42. *Let* $uv \in \mathrm{UDG}(V)$ *with* $\mathrm{HNS}(u, v) = \langle v = w_0, w_1, ..., w_k = w \rangle$, $k \geq 0$, *and let* $y \in \mathrm{D}(u, w)$ *be angle maximizing w.r.t.* uw *whose existence is guaranteed by Lemma 6.39. It holds that*

(i) $\mathrm{d}_{uyv} \geq \mathrm{d}_{uyw}$, *and*

(ii) $\mathrm{C}(u, y, w) \cap \mathrm{H}^v(u, y) \subseteq \mathrm{C}(u, y, v) \cap \mathrm{H}^v(u, y)$.

Proof. For simplicity, let $w_{k+1} := y$. Define area A_i as follows (see Figure 6.11a):

$$A_i := \mathrm{C}(u, y, w_i) \cap \mathrm{H}^v(u, y)\,.$$

The proof relies on the following claim, which is proven first.

Claim. $A_0 \supseteq A_1 \supseteq ... \supseteq A_k$.

Proof. Proof by induction on index i.
I.h.: For an arbitrary but fixed i it holds that $A_0 \supseteq ... \supseteq A_i$.
B.c. ($i = 0$): The base case trivially holds with $A_0 = \mathrm{C}(u, y, w_0) \cap \mathrm{H}^v(u, y)$.
I.s. ($i \to i + 1$): By the i.h. it holds that $A_0 \supseteq ... \supseteq A_i$ and Lemma 6.41 implies that all nodes from $\mathrm{HNS}(u, v)$ are located in the same half plane w.r.t. $\ell(u, y)$.

Assume $w_{i+1} \in \overline{\mathrm{H}^{w_i}(u, w_{i-1})}$. By definition of hidden node sequences, it holds that $w_{i+1} \in \mathrm{D}(u, w_i)$ and $w_i \in \mathrm{D}(u, w_{i-1})$. The latter combined with the assumption that w_{i+1} and w_i are in different half-planes w.r.t. $\ell(u, w_{i-1})$ yields that $w_{i+1} \in \mathrm{D}(u, w_{i-1})$ as well as that $\measuredangle u w_{i+1} w_{i-1} > \measuredangle u w_i w_{i-1}$. But this contradicts the assumption that w_i is the angle maximizing node w.r.t. uw_{i-1} and hence $w_{i+1} \in \mathrm{H}^{w_i}(u, w_{i-1})$ must hold.

Next, define

$$A_{i+1}^o := \mathrm{D}^{\mathrm{o}}(u, y, w_{i+1}) \cap \mathrm{H}^v(u, y)$$

and assume $w_i \in A_{i+1}^o$. Then, using a similar line of argumentation as in the proof of Lemma 6.37 it follows that $\measuredangle u y w_i > \measuredangle u w_{i+1} w_i$, but which contradicts the assumption that w_{i+1} is angle maximizing w.r.t. uw_i. Hence, $w_i \in \overline{A_{i+1}^o}$, $w_{i+1} \in \mathrm{H}^{w_i}(u, w_{i-1})$, and in addition $w_i \notin B := \overline{\mathrm{H}^v(u, y)}$, because all nodes from $\mathrm{HNS}(u, v)$ are in the same half-plane w.r.t. $\ell(u, y)$. This directly implies that $A_i \supseteq A_{i+1}$ and thus, this claim holds. □

By the above claim it holds that $A_0 \supseteq A_k$ and with $w_k = w$ and $w_0 = v$ it follows that

$$\underbrace{\mathrm{C}(u, y, v) \cap \mathrm{H}^v(u, y)}_{A_0} \supseteq \underbrace{\mathrm{C}(u, y, w) \cap \mathrm{H}^v(u, y)}_{A_k},$$

which proves part (ii) of this lemma. It remains to prove part (i).

The fact that $w_{i+1} \in \mathrm{D}(u, w_i)$, for all $0 \leq i < k + 1$, implies

$$\|uy\| = \|uw_{k+1}\| \leq \|uw_k\| \leq \cdots \leq \|uw_0\| = \|uv\|.$$

Partition $\mathrm{C}(u, y, w)$ into two circular sectors using the chord uw (see Figure 6.11b). With $y \in \mathrm{D}(u, w)$ it holds that the diameter of the circular sector containing y is given by $\|uw\| \leq \|uv\| \leq \mathrm{d}_{uyv}$. From $A_0 \supseteq A_k$ it follows that the diameter of the remaining circular sector is at most d_{uyv}. Hence, $\mathrm{d}_{uyv} \geq \mathrm{d}_{uyw}$ and part (i) holds as well. □

Lemma 6.43. *Let $u, v, w \in V$ be nodes such that the diameter of $\mathrm{C}(u, v, w)$ is at most one, i.e., $\mathrm{d}_{uvw} \leq 1$. If there exists $x \in V \setminus \{u, v, w\}$ with $x \in \mathrm{C}(u, v, w)$ and $ux \notin \mathrm{GPDT}(V)$, then there exists a node $\hat{x} \in \mathrm{C}(u, v, w)$ with $u\hat{x} \in \mathrm{GPDT}(V)$.*

Proof. Let $\mathrm{HNS}(u, x) = \langle x = y_0, y_1, ..., y_k = y \rangle$, $k \geq 0$, be the hidden node sequence w.r.t. ux. According to Lemma 6.39, there exists an angle maximizing node $\hat{y} \in \mathrm{D}(u, y)$ w.r.t. uy, s.t. $u\hat{y} \in \mathrm{GPDT}(V)$. If $\hat{y} \in \mathrm{C}(u, w, v)$, then the claim holds with $\hat{x} := \hat{y}$. Thus, assume $\hat{y} \notin \mathrm{C}(u, w, v)$ (see Figure 6.12).

For later reference, it is now first shown that

$$B := \mathrm{C}(u, \hat{y}, y) \cap \overline{\mathrm{H}^{\hat{y}}(u, x)} \subseteq \mathrm{C}(u, w, v).$$

By Lemma 6.42 it holds that $\mathrm{C}(u, \hat{y}, y) \cap \mathrm{H}^x(u, \hat{y}) \subseteq \mathrm{C}(u, \hat{y}, x) \cap \mathrm{H}^x(u, \hat{y})$. This implies in particular that

$$B \subseteq A := \mathrm{C}(u, \hat{y}, x) \cap \overline{\mathrm{H}^{\hat{y}}(u, x)}.$$

Now assume there exists some point $p \in \mathbb{R}^2$, s.t. $p \in A$ and $p \notin \mathrm{D}(u, w, v)$. Then $\mathrm{C}(u, w, v)$ and $\mathrm{C}(u, \hat{y}, x)$ intersect four times, since $\hat{y}, p \notin \mathrm{C}(u, w, v)$, $x \in \mathrm{C}(u, w, v)$, but $\hat{y}$ and p are

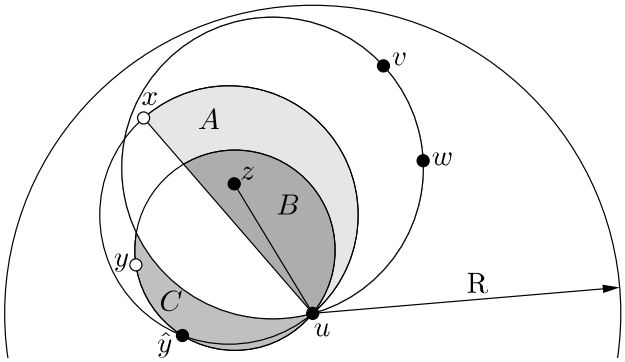


Figure 6.12: Illustration of areas required for the proof of Lemma 6.43. White nodes are hidden nodes.

located on different sides of the straight line $\ell(u, x)$. However, the number of intersections of two circles in the Euclidean plane is either at most two, or infinity. Hence, the above case cannot occur and $A \subseteq \mathrm{C}(u, w, v)$ must hold, which implies $B \subseteq A \subseteq \mathrm{C}(u, w, v)$.

Observe that uy cannot violate the second PDT criterion, since B is entirely contained by $\mathrm{C}(u, v, w)$, whose diameter is $\mathrm{d}_{uvw} \leq 1$. In combination with Lemma 6.31, $\mathrm{d}_{u\hat{y}y} \leq 1$ follows. However, because $uy \notin \mathrm{GPDT}(V)$, uy must violate the first PDT criterion and thus, there must exist $z \in \mathrm{C}(u, \hat{y}, y)$. In accordance with Lemma 6.40, there cannot exist a node within $\mathrm{D}^{\mathrm{o}}(u, \hat{y}, y) \cap \mathrm{H}^{\hat{y}}(u, x)$ and therefore, $z \in B$ and in particular $z \in \mathrm{C}(u, w, v)$ must hold.

If $uz \in \mathrm{GPDT}(V)$, the claim holds with $\hat{x} := z$. Otherwise, repeat the above line of argumentation inductively with z instead of x and circle $\mathrm{C}(u, \hat{y}, y)$ instead of circle $\mathrm{C}(u, v, w)$.

Note that each node which is contained by $\mathrm{C}(u, \hat{y}, y)$ must also be contained by $\mathrm{C}(u, w, v)$, because Lemma 6.40 particularly implies that area C (see Figure 6.12)

$$C := \overline{\mathrm{C}(u, v, w)} \cap \mathrm{C}(u, \hat{y}, y)$$

is empty of nodes from V. Furthermore, note that this induction eventually terminates, because $x \notin \mathrm{C}(u, \hat{y}, y)$ (which is due to the fact that $B \subseteq A$ and the assumption that no four points in V are cocircular) cannot take over the role of z in the induction step. This implies in particular that the number of nodes (candidates) which could possibly play z's role shrinks by at least one in the induction step. Now assume there would be no candidate left after the induction's termination. Then, uy would neither violate the first nor the second PDT criterion implying $uy \in \mathrm{GPDT}(V)$. But this contradicts the initial assumption that $\mathrm{HNS}(u, v) = \langle v = y_0, y_1, ..., y_k = y \rangle$ is a hidden node sequence containing node y. That is, the only possibility of the induction's termination without finding a GPDT neighbor of u within $\mathrm{C}(u, w, v)$ results in a contradiction. From this it can finally be concluded that such a GPDT neighbor of u must exist in $\mathrm{C}(u, w, v)$. □

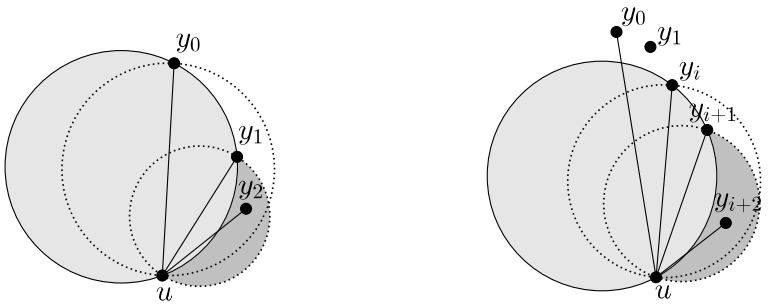


Figure 6.13: Illustration for the proof of Lemma 6.44. (a) Base case and (b) inductive step. Gabriel circles are dotted.

Lemma 6.44. *Let $u \in V$ be any node and let $\langle y_0, y_1, ..., y_k \rangle$ be a sequence of nodes from V, s.t. $uy_0 \in \mathrm{GPDT}(V)$ and for all $0 \leq i < k$, $y_{i+1} \in \mathrm{D}(u, y_i)$ is angle maximizing w.r.t. uy_i. Then, $y_k \in \mathrm{GPDT}(V)$ holds.*

Proof. The main observation of this proof is that all edges uy_i are contained in $\mathrm{GPDT}(V)$. This is proven by induction on index i. The induction hypothesis is tripartite: Parts (ii) and (iii) correspond to the first and the second GPDT criterion, respectively, whereas (i) just serves to prove the other two. Part (i) represents the fact that all nodes y_i are located in the same half-plane w.r.t. uy_0, as illustrated in Figure 6.13b.
I.h.: For an arbitrary but fixed i, the following holds:

(i) $y_{j+1} \in \overline{\mathrm{H}^{y_0}(u, y_j)}$, for all $1 \leq j \leq i$,

(ii) $\mathrm{C}(u, y_1, y_0) \cap \overline{\mathrm{H}^{y_1}(u, y_0)} \supseteq \ldots \supseteq \mathrm{C}(u, y_{i+1}, y_i) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$, and

(iii) $\mathrm{d}_{uy_1y_0} \geq \mathrm{d}_{uy_2y_1} \geq \ldots \geq \mathrm{d}_{uy_{i+1}y_i}$.

B.c. ($i = 1$): From Lemma 6.36 and the assumption that $uy_0 \in \mathrm{GPDT}(V)$ it immediately follows that $\mathrm{D}^{\mathrm{o}}(u, y_1, y_0)$ is empty of nodes from V. By assumption, $y_2 \in \mathrm{D}(u, y_1)$ and hence it can only be contained in $\mathrm{D}(u, y_1) \setminus \mathrm{C}(u, y_1, y_0)$ (dark gray area in Figure 6.13a). Hence, $y_2 \in \overline{\mathrm{H}^{y_0}(u, y_1)}$ and part (i) of the i.h. holds. Moreover, from the observation it follows that

$$\mathrm{C}(u, y_2, y_1) \cap \overline{\mathrm{H}^{y_1}(u, y_0)} \subseteq \mathrm{C}(u, y_1, y_0) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}\,,$$

which in turn implies that $\mathrm{d}_{uy_1y_0} \geq \mathrm{d}_{uy_2y_1}$. Hence, part (ii) and (iii) hold for $i = 1$.
I.s. ($i \to i + 1$): By assumption $uy_0 \in \mathrm{GPDT}(V)$ and therefore, $\mathrm{C}(u, y_1, y_0) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$ is empty of nodes from V. With part (ii) of the i.h. this also holds for $\mathrm{C}(u, y_{i+1}, y_i) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$. Furthermore, Lemma 6.40 implies that $\mathrm{D}^{\mathrm{o}}(u, y_{i+1}, y_i) \cap \mathrm{H}^{y_{i+1}}(u, y_0)$ is empty of nodes from V. Moreover, $\mathrm{H}^{y_{i+1}}(u, y_0) = \mathrm{H}^{y_1}(u, y_0)$ since y_1 and y_{i+1} are on the same side w.r.t. the straight line $\ell(u, y_0)$ according to part (1) of the i.h. Using this equivalence, it holds that $\mathrm{D}^{\mathrm{o}}(u, y_{i+1}, y_i) \cap \mathrm{H}^{y_1}(u, y_0)$ is empty of nodes from V and hence, $\mathrm{D}^{\mathrm{o}}(u, y_{i+1}, y_i)$ is empty of nodes.

By assumption $y_{i+2} \in \mathrm{D}(u, y_{i+1})$ and thus it can only be contained in $\mathrm{D}(u, y_{i+1}) \setminus \mathrm{D}^{\mathrm{o}}(u, y_{i+1}, y_i)$ (dark gray area in Figure 6.13b). Hence, $y_{i+2} \in \overline{\mathrm{H}^{y_0}(u, y_{i+1})}$ and part (i) of the i.h. holds.

Moreover, from the observation it follows that

$$\mathrm{C}(u, y_{i+2}, y_{i+1}) \cap \overline{\mathrm{H}^{y_1}(u, y_0)} \subseteq \mathrm{C}(u, y_{i+1}, y_i) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}\,,$$

which in turn implies that $\mathrm{d}_{uy_{i+1}y_i} \geq \mathrm{d}_{uy_{i+2}y_{i+1}}$. Hence, part (ii) and (iii) of the i.h. hold for $i + 1$.

Termination: Considering edge uy_k, two cases may occur.

Either there exists an angle maximizing node $y_{k+1} \in \mathrm{D}(u, y_k)$, or not. In the latter case $uy_k \in \mathrm{GPDT}(V)$ holds trivially. Therefore, assume the former holds.

Because $uy_0 \in \mathrm{GPDT}(V)$, it holds that $\mathrm{C}(u, y_1, y_0) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$ is empty of nodes from V. Applying i.h. (ii) yields that $\mathrm{C}(u, y_{k+1}, y_k) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$ is also empty of nodes from V. Now, from i.h. part (i) it follows that $y_2 \in \overline{\mathrm{H}^{y_0}(u, y_1)}$, $y_3 \in \overline{\mathrm{H}^{y_0}(u, y_2)}$, up to $y_{k+1} \in \overline{\mathrm{H}^{y_0}(u, y_k)}$. This implies that all nodes $y_1, y_2, y_3, \ldots, y_{k+1}$ are located in the same half-plane w.r.t. uy_0. Because especially y_1 and y_{k+1} are in the same half-plane, it follows that $\mathrm{C}(u, y_{k+1}, y_k) \cap \overline{\mathrm{H}^{y_{k+1}}(u, y_0)} = \mathrm{C}(u, y_{k+1}, y_k) \cap \overline{\mathrm{H}^{y_1}(u, y_0)}$ holds.

According to Lemma 6.40, it follows that the empty area $\mathrm{C}(u, y_{k+1}, y_k) \cap \overline{\mathrm{H}^{y_{k+1}}(u, y_0)}$ can be extended to $\mathrm{C}(u, y_{k+1}, y_k) \cap \overline{\mathrm{H}^{y_{k+1}}(u, y_k)}$ and is still empty of nodes from V. Therefore, uy_k cannot violate the first GPDT criterion. According to i.h. part (iii) and the fact that $uy_0 \in \mathrm{GPDT}(V)$, it follows that $\mathrm{d}_{uy_{k+1}y_k} \leq \mathrm{d}_{uy_1y_0} \leq 1$. Thus, uy_k is also not violating the second GPDT criterion and therefore, $uy_k \in \mathrm{GPDT}(V)$. □

Lemma 6.45. *Given four nodes $u, v, w, x \in V$, s.t. $\mathrm{d}_{uvw} \leq 1$ and $x \in \mathrm{C}(u, v, w)$, then there exists a node $\hat{x} \in V$ satisfying the following three properties:*

(i) $u\hat{x} \in \mathrm{GPDT}(V)$,

(ii) $\hat{x} \in \mathrm{C}(u, v, w)$, *and*

(iii) $\mathrm{d}_{u\hat{y}\hat{x}} \leq \mathrm{d}_{uvw}$, *if there exists an angle maximizing node $\hat{y} \in \mathrm{D}(u, \hat{x})$ w.r.t. $u\hat{x}$.*

Proof. See Figure 6.14 for an illustration of this proof. Edge ux may or may not be contained in $\mathrm{GPDT}(V)$. If $ux \notin \mathrm{GPDT}(V)$, then replace x by the node x' whose existence is guaranteed by Lemma 6.43 and for which it holds that $x' \in \mathrm{C}(u, v, w)$ and $ux' \in \mathrm{GPDT}(V)$.

Consider the longest possible sequence $S := \langle x = y_0, y_1, ..., y_k \rangle$ s.t. $y_{i+1} \in \mathrm{D}(u, y_i)$, $0 \leq i < k$, is angle maximizing w.r.t. uy_i and $y_k \in \mathrm{C}(u, v, w)$ holds. Apply Lemma 6.44 on sequence S and observe that $uy_k \in \mathrm{GPDT}(V)$ holds. With $\hat{x} := y_k$, part (i) and part (ii) of the claim immediately hold.

Now, assume there exists an angle maximizing node $\hat{y} \in \mathrm{D}(u, \hat{x})$ w.r.t. $u\hat{x}$ (otherwise, the proof is finished). Then $\hat{y} \notin \mathrm{C}(u, v, w)$ must hold, because otherwise $S \cup \{\hat{y}\}$ would be a valid but longer sequence than S, contradicting the maximality of S. Finally, since $\hat{y} \notin \mathrm{C}(u, v, w)$, $\hat{y} \in \mathrm{D}(u, \hat{x})$, and $\hat{x} \in \mathrm{C}(u, v, w)$, it follows that $\mathrm{d}_{u\hat{y}\hat{x}} \leq \mathrm{d}_{uvw}$. □

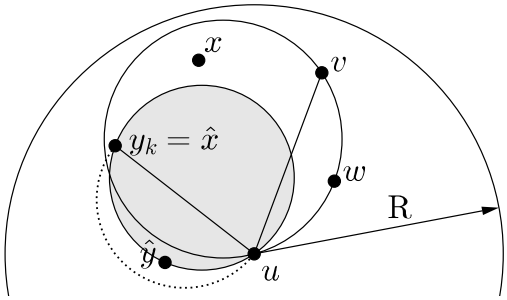


Figure 6.14: Illustration for the proof of Lemma 6.45. Circle $\mathrm{C}(u, v, w)$ is fully contained by the unit disk centered at u. The angle maximizing node $\hat{y}$ w.r.t. uy_k is outside of $\mathrm{C}(u, v, w)$. The dotted circle represents the Gabriel circle.

In the following, for any unit disk graph edge uv, define the *angle maximizing* GPDT *node* w.r.t. uv as the node $w \in \mathrm{D}(u, v)$ with $uw \in \mathrm{GPDT}(V)$ which maximizes the angle $\measuredangle uwv$ among all such nodes in $\mathrm{D}(u, v)$. Moreover, $\mathrm{expire}_{\mathrm{t}(v)}$ refers to the maximum duration of this node's timer $\mathrm{t}(v)$ during algorithm execution. That is, for two nodes $u, v \in$ that start their timers upon reception of an RTS, if the timer of v expires later than the timer of w, then $\mathrm{expire}_{\mathrm{t}(v)} > \mathrm{expire}_{\mathrm{t}(w)}$. This is used to compare ultimate timeouts of nodes in the following proofs.

Lemma 6.46. *Let $uv \in \mathrm{UDG}(V)$ be an arbitrary edge and let $w \in \mathrm{D}(u, v)$ be the angle maximizing* GPDT *node w.r.t. uv. It holds that timer $\mathrm{t}(w)$ expires earlier than timer $\mathrm{t}(v)$, i.e., $\mathrm{expire}_{\mathrm{t}(w)} < \mathrm{expire}_{\mathrm{t}(v)}$.*

Proof. Let sequence $\langle v_1, v_2, ..., v_k \rangle$ represent the distance ordered neighborhood of u, i.e, $\|uv_i\| \leq \|uv_{i+1}\|$, for all $1 \leq i < k$. Assume w.l.o.g. that $\mathrm{t}_{\max} = 1$.
I.h.: For an arbitrary but fixed i and all $1 \leq j \leq i$, the following holds: If there exists an angle maximizing GPDT node $w \in \mathrm{D}(u, v_j)$, then $\mathrm{expire}_{\mathrm{t}(w)} < \mathrm{expire}_{\mathrm{t}(v_j)}$.
B.c. ($i = 1$): I.h. for $i = 1$ holds trivially, because there is no node closer to u than v_1.
I.s. ($i \to i + 1$): If $\mathrm{D}(u, v_{i+1})$ does not contain any GPDT node, the claim holds trivially. Therefore, let $w \in \mathrm{D}(u, v_{i+1})$ be the angle maximizing GPDT node.

If $\mathrm{D}(u, w)$ does not contain any GPDT node, then during execution of algorithm rPDT, $\mathrm{expire}_{\mathrm{t}(w)} = \|uw\|$. With $w \in \mathrm{D}(u, v_{i+1})$ follows that

$$\mathrm{expire}_{\mathrm{t}(v_{i+1})} \geq \|uv_{i+1}\| > \|uw\| = \mathrm{expire}_{\mathrm{t}(w)} .$$

Otherwise, let $\hat{w} \in \mathrm{D}(u, w)$ be the angle maximizing GPDT node w.r.t. uw. Two cases can be distinguished: (i) $\hat{w} \notin \mathrm{D}(u, v_{i+1})$ and (ii) $\hat{w} \in \mathrm{D}(u, v_{i+1})$.

Case (i)—see Figure 6.15a: With $w \in \mathrm{D}(u, v_{i+1})$ and $\hat{w} \in \mathrm{D}(u, w)$, it follows that $\|uv_{i+1}\| > \mathrm{d}_{u\hat{w}w}$ and $\|uv_{i+1}\| > \|uw\|$. Because $\hat{w}$ is the angle maximizing GPDT node w.r.t. uw, during execution of rPDT, the timeout of timer $\mathrm{t}(w)$ can be increased to at most $\mathrm{d}_{u\hat{w}w}$ and it holds that $\mathrm{expire}_{\mathrm{t}(w)} \leq \max\{\mathrm{d}_{u\hat{w}w}, \|uw\|\}$. The latter implies

$$\mathrm{expire}_{\mathrm{t}(v_{i+1})} \geq \|uv_{i+1}\| > \max\{\mathrm{d}_{u\hat{w}w}, \|uw\|\} \geq \mathrm{expire}_{\mathrm{t}(w)} .$$

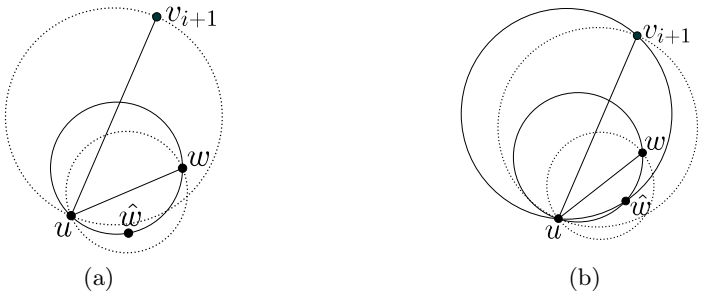


Figure 6.15: Illustrations for the proof of Lemma 6.46. (a) Case (i), where $\hat{w} \notin \mathrm{D}(u, v_{i+1})$. (b) Case (ii), where $\hat{w} \in \mathrm{D}(u, v_{i+1})$. The dotted circles represent $\mathrm{D}(u, w)$ and $\mathrm{D}(u, v_{i+1})$, whereas the bold circles represent $\mathrm{C}(u, \hat{w}, w)$ and $\mathrm{C}(u, \hat{w}, v_{i+1})$.

Case (ii)—see Figure 6.15b: Because $\|uw\| < \|uv_{i+1}\|$ and $\hat{w}$ is the angle maximizing GPDT node w.r.t. uw, the i.h. implies that $\mathrm{expire}_{\mathrm{t}(\hat{w})} < \mathrm{expire}_{\mathrm{t}(w)}$. Then, during execution of rPDT, $\hat{w}$ sends a CTS message before timer $\mathrm{t}(w)$ expires. Because $\hat{w} \in \mathrm{D}(u, v_{i+1})$, during algorithm execution $\mathrm{timeout}_{\mathrm{t}(v_{i+1})}$ will either be set to $\mathrm{d}_{u\hat{w}v_{i+1}}$, or is is already at least as large before timer $\mathrm{t}(v_{i+1})$ expires. Now, because $w \in \mathrm{D}(u, v_{i+1})$ and $\hat{w} \in \mathrm{D}(u, w)$ are the angle maximizing GPDT nodes w.r.t. uv_{i+1} and uw, respectively, Lemma 6.37 implies that $\mathrm{d}_{u\hat{w}w} < \mathrm{d}_{u\hat{w}v_{i+1}}$ (where the strict inequality holds due to the non-cocircularity assumption). It follows that

$$\mathrm{expire}_{\mathrm{t}(v_{i+1})} \geq \mathrm{d}_{u\hat{w}v_{i+1}} > \mathrm{d}_{u\hat{w}w} = \max\{\mathrm{d}_{u\hat{w}w}, \|uw\|\} \geq \mathrm{expire}_{\mathrm{t}(w)}\,.$$

Hence, the i.h. holds for $i+1$ in all possible cases. Since the proof holds for each node v_i in the ordered neighborhood of u, it holds in particular for any particular node v as claimed in the Lemma. □

Theorem 6.47. *Let* $\mathrm{UDG}(V)$ *be any unit disk graph over a distinct and finite node set* $V \subset \mathbb{R}^2$*, which is neither cocircular nor collinear. During the execution of algorithm rPDT initiated by any node* $u \in V$*, within time* $\mathrm{t}_{\max}$*, each neighbor* $v \in \mathrm{N}_1(u)$ *sends a* CTS *message, if and only if* $uv \in \mathrm{GPDT}(V)$*.*

Proof. Let $v \in \mathrm{N}_1(u)$ be an arbitrary node and assume w.l.o.g. that $\mathrm{t}_{\max} = 1$. The two implications of the logic equivalence are shown separately as follows:

"⇒" v sends a CTS message within time $\mathrm{t}_{\max}$ if $uv \in \mathrm{GPDT}(V)$, and

"⇐" v does not send a CTS message at all if $uv \notin \mathrm{GPDT}(V)$.

"⇒" Because $uv \in \mathrm{GPDT}(V)$, during algorithm execution v does not detect any violation of the GPDT criteria. Thus, v's timer expires eventually and it sends a CTS. However, it has to be proven that sending of the CTS happens after time at most $\mathrm{t}_{\max}$ past sending of the RTS by the initiating node u.

It holds that $\mathrm{expire}_{\mathrm{t}(v)} = \max\{\mathrm{d}_{uwv}, \|uv\|\}$, where d_{uwv} is the diameter of $\mathrm{C}(u, w, v)$ with w being the node maximizing d_{uwv} among all nodes contained in $\mathrm{D}(u, v)$ that send

a CTS. If $\mathrm{d}_{uwv} > 1$ holds, then uv violates the second GPDT criterion with angle maximizing node w, which contradicts to uv being contained in $\mathrm{GPDT}(V)$. Moreover, with $v \in \mathrm{N}_1(u)$, it holds that $\|uv\| \leq 1$ and thus, $\mathrm{expire}_{\mathrm{t}(v)} = \max\{\mathrm{d}_{uwv}, \|uv\|\} \leq 1$. I.e., v sends a CTS no later than one time unit past sending of the RTS by node u, which proves the first part of the theorem.

"$\Leftarrow$" In this case $uv \notin \mathrm{GPDT}(V)$ holds. Let $\mathrm{HNS}(u,v) = \langle v = w_0, w_1, ..., w_k = w\rangle$, $k \geq 0$, be the hidden node sequence w.r.t. uv.

According to Lemma 6.39, there exists an angle maximizing node $y \in \mathrm{D}(u,w)$ w.r.t. uw s.t. $uy \in \mathrm{GPDT}(V)$ holds.

If there is more than one hidden node sequence, choose the sequence s.t. y maximizes the angle $\measuredangle uyv$, i.e., choose the hidden node sequence for which node y is the angle maximizing GPDT node w.r.t. uv.

Node y may or may not be contained in $\mathrm{D}(u,v)$. In the following, these cases are distinguished and proven separately. For each case it is required to show

1. the existence of nodes that witness the violation of at least one of the GPDT criteria for edge uv,
2. the visibility of the aforementioned nodes, i.e., that these nodes are GPDT nodes and their timers expire earlier than the timer of v, and
3. that these nodes are neighbors of v in $\mathrm{UDG}(V)$.

The first property is needed for v being able to detect a violation of the GPDT criteria during algorithm execution. The second property ensures that these nodes send a CTS message timely. Finally, the third property ensures that v can actually overhear these CTS messages and are therefore contained in v's set of known neighbors $\mathrm{S}(v)$.

Case 1 ($y \in \mathrm{D}(u,v)$): Node w might be hidden because uw either violates the first or the second PDT criterion (if both criteria are violated, consider the violation of the second criterion, which is handled in Case 1.1).

Case 1.1: Assume uw violates the second PDT criterion, as given in Figure 6.16. Then it holds that $\sin(\beta) < \|uw\|$, where $\beta = \measuredangle uyw$. By Lemma 6.30, $\mathrm{d}_{uyw} = \|uw\|/\sin(\beta)$ and $\mathrm{d}_{uyv} = \|uv\|/\sin(\delta)$, where $\delta = \measuredangle uyv$. Property (i) of Lemma 6.42 implies that $\mathrm{d}_{uyv} \geq \mathrm{d}_{uyw}$ and thus,

$$\begin{aligned} & \|uv\|/\sin(\delta) = \mathrm{d}_{uyv} \geq \mathrm{d}_{uyw} = \|uw\|/\sin(\beta) > 1 \\ \Leftrightarrow \quad & \|uv\| > \sin(\delta)\,. \end{aligned}$$

That is, uv violates the second GPDT criterion with angular node y. By assumption $y \in \mathrm{D}(u,v)$ and by construction and choice of $\mathrm{HNS}(u,v)$ it holds that y is the angle maximizing GPDT node w.r.t. uv. The application of Lemma 6.46 yields that timer $\mathrm{t}(y)$ expires earlier than timer $\mathrm{t}(v)$. Thus, y sends a CTS before expiration of timer $t(v)$. With $y \in \mathrm{D}(u,v)$ and $uv \in \mathrm{UDG}(V)$ it holds that $vy \in \mathrm{UDG}(V)$ and hence, v actually overhears the CTS from y and adds it to its set of known neighbors $\mathrm{S}(v)$. Therefore, during algorithm execution, node v is able to detect a violation of the second GPDT criterion for uv with angular node $y \in \mathrm{S}(v)$.

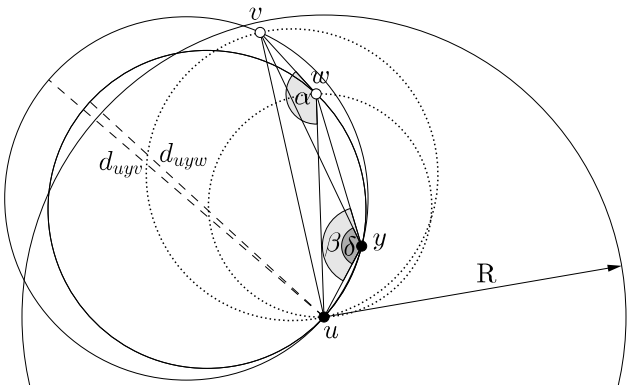


Figure 6.16: Illustration for the proof of Theorem 6.47, Case 1.1. Node y sends a CTS before expiration of timer $\mathrm{t}(v)$ and serves as a witness that $uv \notin \mathrm{GPDT}(V)$. Gabriel circles are represented by dotted circles. The dashed lines represent diameters. White nodes are hidden nodes.

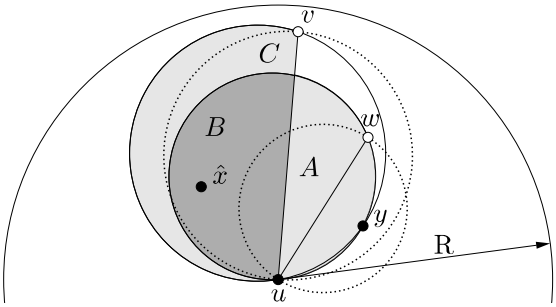


Figure 6.17: Illustration for proof of Theorem 6.47, Case 1.2. Edge uw violates the first PDT criterion with angular node y and witness $\hat{x}$. Gabriel circles are dotted. White nodes are hidden nodes.

Case 1.2: Assume uw violates the first PDT criterion, as given in Figure 6.17. Define areas A, B, and C as follows:

$$\begin{aligned} A &:= \mathrm{C}(u,y,w) \cap \mathrm{H}^y(u,v) \\ B &:= \mathrm{C}(u,y,w) \cap \overline{\mathrm{H}^y(u,v)} \\ C &:= \mathrm{C}(u,y,v) \cap \overline{\mathrm{H}^y(u,v)} \end{aligned}$$

Because uw violates the first PDT criterion with angle maximizing node y, there exists at least one node $x \in \mathrm{C}(u,y,w)$. Since cases where in addition the second PDT criterion is violated are handled by Case 1.1, it holds that $\mathrm{d}_{uyw} \leq 1$.

The application of Lemma 6.45 on $\mathrm{C}(u,y,w)$ and x guarantees the existence of a node $\hat{x} \in \mathrm{C}(u,y,w)$ with $u\hat{x} \in \mathrm{GPDT}(V)$ and $\mathrm{d}_{u\hat{y}\hat{x}} \leq \mathrm{d}_{uyw}$ (if there exists an angle maximizing node $\hat{y} \in \mathrm{D}(u,\hat{x})$ w.r.t. $u\hat{x}$).

Lemma 6.40 implies that area A is empty of nodes and hence, $\hat{x} \in B$ must hold. Additionally, by Lemma 6.42 $B \subseteq C$ and hence, $\hat{x} \in C$ holds. This implies that uv

violates the first GPDT criterion with angular node y and witness $\hat{x}$. Lemma 6.42 yields in addition that $\mathrm{d}_{uyw} < \mathrm{d}_{uyv}$ (where the strict inequality holds due to the non-corcirularity assumption).

If the angular maximizing node $\hat{y} \in \mathrm{D}(u, \hat{x})$ exists, then

$$\mathrm{expire}_{\mathrm{t}(\hat{x})} \leq \mathrm{d}_{u\hat{y}\hat{x}} \overset{\text{Lemma 6.45}}{\leq} \mathrm{d}_{uyw} \overset{\text{Lemma 6.42}}{<} \mathrm{d}_{uyv} \leq \mathrm{expire}_{\mathrm{t}(v)} .$$

Otherwise, if $\hat{y}$ does not exist, it follows that

$$\mathrm{expire}_{\mathrm{t}(\hat{x})} = \|u\hat{x}\| \overset{(1)}{<} \mathrm{d}_{uyw} \overset{\text{Lemma 6.42}}{<} \mathrm{d}_{uyv} \leq \mathrm{expire}_{\mathrm{t}(v)} ,$$

where (1) follows from the fact that $\hat{x} \in \mathrm{C}(u, y, w)$. In summary, in both cases timer $\mathrm{t}(\hat{x})$ expires earlier than timer $\mathrm{t}(v)$. In addition, with $u\hat{x} \in \mathrm{GPDT}(V)$, $\hat{x} \in C \subseteq \mathrm{C}(u, y, v)$, and $\mathrm{d}_{uyv} \leq 1$ it holds that v actually overhears $\hat{x}$'s CTS and therefore $\hat{x} \in \mathrm{S}(v)$ holds before the expiration of timer $\mathrm{t}(v)$.

Now consider node y. Because $y \in \mathrm{D}(u, v)$ and y is the angle maximizing GPDT node w.r.t. uv, by Lemma 6.46 timer $\mathrm{t}(y)$ expires earlier than timer $\mathrm{t}(v)$, i.e., y sends a CTS before timer $\mathrm{t}(v)$ expires. Furthermore, with $y \in \mathrm{D}(u, v)$ and $uv \in \mathrm{UDG}(V)$ it holds that $vy \in \mathrm{UDG}(V)$ and hence, v actually overhears the CTS from y and adds it to its set of known neighbors $\mathrm{S}(v)$ before expiration of its timer $\mathrm{t}(v)$.

Thus, at the time of expiration of timer $\mathrm{t}(v)$, $\{\hat{x}, y\} \subseteq \mathrm{S}(v)$ holds and v can detect a violation of the first GPDT criterion with angular node y and witness node $\hat{x}$. According to algorithm, rPDT, node v refrains from sending a CTS and remains silent.

Case 2 ($y \notin \mathrm{D}(u, v)$): The following proof shows that this case leads to a contradiction and therefore cannot occur. For simplicity, let $w_{k+1} := y$.

Among all nodes in $\mathrm{HNS}(u, v) \cup \{w_{k+1}\} = \langle v = w_0, w_1, \ldots, w_k, w_{k+1} = y \rangle$, let l be the smallest index such that w_l is located outside of $\mathrm{D}(u, v)$. Then node w_{l-1} is a hidden node, located in $\mathrm{D}(u, v)$.

In the following case distinction it is proven that uw_{l-1} can neither violate the first PDT criterion (Case 2.2), nor the second PDT criterion (Case 2.1), which contradicts to the assumption that $w_{l-1} \in \mathrm{HNS}(u, v)$ is a hidden node. That is, the assumption that $y \notin \mathrm{D}(u, v)$ leads to a contradiction and hence, $y \in \mathrm{D}(u, v)$ must hold.

Case 2.1: Assume uw_{l-1} violates the second PDT criterion.

Recall that $w_{l-1} \in \mathrm{D}(u, v)$, whereas w_l is assumed to be strictly outside $\mathrm{D}(u, v)$. Therefore, $\mathrm{D}(u, v) \neq \mathrm{C}(u, w_{k+1}, w_{l-1})$ and these circles intersect exactly twice in area $A := \mathrm{H}^{w_l}(u, w_{l-1})$, as given in Figure 6.18.

In order to violate the second PDT criterion, it must hold that $\sin(\beta) < \|uw_{l-1}\|$, $\beta = \measuredangle uw_l w_{l-1}$, which implies according to Lemma 6.31 that $\mathrm{d}_{uw_l w_{l-1}} > 1$, i.e., $\mathrm{C}(u, w_l, w_{l-1})$ exceeds the unit disk centered at u in $\overline{A} = \overline{\mathrm{H}^{w_l}(u, w_{l-1})}$. Then, there exists a point $p \in \mathbb{R}^2$ with $p \in \mathrm{C}(u, w_l, w_{l-1}) \cap \overline{A}$. Because $\mathrm{D}(u, v)$ is fully contained by the unit disk centered at u, it holds that $p \notin \mathrm{D}(u, v)$. But then, because both u and w_{l-1} are contained in $\mathrm{D}(u, v)$, $\mathrm{C}(u, w_l, w_{l-1})$ and $\mathrm{D}(u, v)$ must also intersect in area $\overline{A}$, which leads to more than two points of intersection. Since $\mathrm{D}(u, v) \neq \mathrm{C}(u, w_{k+1}, w_{l-1})$, this is a contradiction

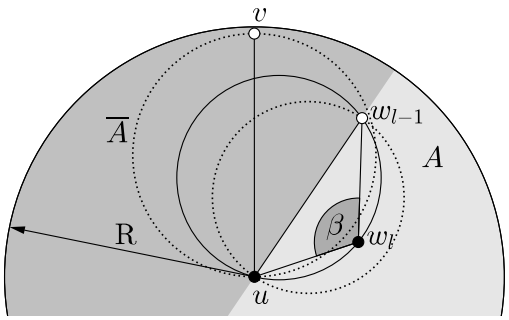


Figure 6.18: Illustration for the proof of Theorem 6.47, Case 2.1. Edge uw_{l-1} violates the second PDT criterion with angle maximizing node w_l. Gabriel circles are dotted and hidden nodes are white.

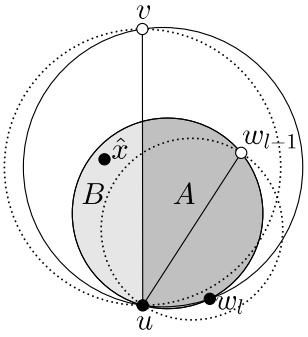


Figure 6.19: Illustration for the proof of Theorem 6.47, Case 2.2. Edge uw_{l-1} violates the first PDT criterion with angle maximizing node w_l and witness $\hat{x}$. Gabriel circles are dotted and white nodes represent hidden nodes.

to the fact that two non-identical circles in the plane can intersect at most two times. Thus, it cannot be the case that uw_{l-1} violates the second PDT criterion.

Case 2.2: Assume uw_{l-1} violates the first PDT criterion, but not the second one as such cases are handled by Case 2.1.

Define areas A and B as follows, and see Figure 6.19 for an illustration.

$$\begin{aligned} A &:= \mathrm{C}(u, w_l, w_{l-1}) \cap \mathrm{H}^{w_l}(u, v) \\ B &:= \mathrm{C}(u, w_l, w_{l-1}) \cap \overline{\mathrm{H}^{w_l}(u, v)} \end{aligned}$$

Because uw_{l-1} violates the first PDT criterion with angle maximizing node w_l, there exists at least one node $x \in \mathrm{C}(u, w_l, w_{l-1})$. Since uw_{l-1} does not violate the second PDT criterion it holds that $\mathrm{d}_{uw_lw_{l-1}} \leq 1$.

The application of Lemma 6.45 on $\mathrm{C}(u, w_l, w_{l-1})$ and node x then guarantees the existence of a node $\hat{x} \in \mathrm{C}(u, w_l, w_{l-1})$ with $u\hat{x} \in \mathrm{GPDT}(V)$. By Lemma 6.40, area A is empty of nodes from V. Since V is non-collinear, $\hat{x} \in B$ follows.

Next, it is shown by contradiction that $\hat{x} \in \mathrm{D}(u, v)$ holds.

Assume $\hat{x} \notin \mathrm{D}(u,v)$. Recall that w_l and $\hat{x}$ lie on different sides of the straight line $\ell(u,v)$. Because $w_l \notin \mathrm{D}(u,v)$ and $u, w_{l-1} \in \mathrm{D}(u,v)$, $\mathrm{D}(u,v)$ intersects $\mathrm{C}(u,w_l,w_{l-1})$ twice in A. Because $\hat{x} \in B$ and $\hat{x}$ is assumed to be outside of $\mathrm{D}(u,v)$, $\mathrm{D}(u,v)$ intersects $\mathrm{C}(u,w_l,w_{l-1})$ two more times in B. In combination with the observation that $\mathrm{D}(u,v) \neq \mathrm{C}(u,w_l,w_{l-1})$, this is a contradiction, since any two non-identical circles in the plane can intersect at most two times. From this it can be concluded that $\hat{x} \in \mathrm{D}(u,v)$ holds, which implies that $\measuredangle u\hat{x}v \geq \pi/2$.

Recall that w_{l-1} is the last node in $\mathrm{HNS}(u,v) \cup \{y = w_{k+1}\}$ being contained in $\mathrm{D}(u,v)$. This implies in particular that $y \notin \mathrm{D}(u,v)$. Thus, $\measuredangle uyv < \pi/2$ holds. But this contradicts the choice of $\mathrm{HNS}(u,v) \cup \{y\}$, which was chosen among all possible sequences such that y is the angle maximizing GPDT node w.r.t. uv.

In summary, Case 2 leads to contradictions and therefore only Case 1 may actually apply, for which the theorem has already been proven to hold. □

6.2.3 Discussion

First of all it can be observed that algorithm rPDT is extremely efficient regarding message complexity and message size.

Theorem 6.48. *During execution of algorithm rPDT, initiated by any node $u \in \mathrm{UDG}(V)$, exactly $|\mathrm{N}_1^{\mathrm{PDT}(V)}(u)|$ messages are being transmitted, where $\mathrm{N}_1^{\mathrm{PDT}(V)}(u)$ denotes the one-hop neighborhood of u in $\mathrm{PDT}(V)$, including u itself. Moreover, each such message is of size $\mathcal{O}(\mathrm{P}_{\max})$ bits, where $\mathrm{P}_{\max}$ is the number of bits required to represent a single node position.*

Proof. By Theorem 6.47, exactly all those nodes $v \in \mathrm{N}_1(u)$ send a CTS message for which $uv \in \mathrm{GPDT}(V)$ holds. Since $\mathrm{GPDT}(V) = \mathrm{PDT}(V)$ by Theorem 6.35, it can be equivalently stated that all neighbors of u in $\mathrm{PDT}(V)$ reply with a CTS message. Moreover, node u sends one initial RTS, which gives $|\mathrm{N}_1^{\mathrm{PDT}(V)}(u)|$ message transmissions in total. Furthermore, RTS and CTS messages contain only a single node position and hence, the message size is $\mathcal{O}(\mathrm{P}_{\max})$ bits. □

Under the assumption that all nodes are completely unaware of their network neighborhoods at algorithm start, algorithm rPDT is in fact *message optimal*, at least in the following sense. In order for node u to initiate some message passing routine, it has to transmit at least one message. For node u to discover the existence of all of its PDT neighbors, each such neighbor has to transmit at least one message in order to announce its existence. Hence, at least $|\mathrm{N}_1^{\mathrm{PDT}(V)}(u)|$ messages have to be transmitted, and this coincides with the upper bound message complexity of the algorithm.

The above observations are strongly connected to the next result.

Theorem 6.49. *rPDT is an $\mathcal{O}$-reactive local view topology control algorithm.*

Proof. Under the assumption that $\mathcal{O}(\mathrm{P}_{\max}) \in \mathcal{O}(\log n)$, it holds by Theorem 6.48 that the number of bits transmitted during execution of algorithm rPDT by any node $v \in \mathrm{UDG}(V)$

is $\mathcal{O}(|\mathrm{PDT}(\mathrm{UDG}(V))[v]| \cdot \log n)$ and therefore, rPDT is an $\mathcal{O}$-reactive local view topology control algorithm. □

To the best of the author's knowledge, algorithm rPDT is the first reactive local view topology control algorithm for construction of planar, constant stretch Euclidean spanners. This positively answers the previously raised question (see e.g., [14] and [15]), if this is possible at all. Moreover, it is the first such algorithm that does not require sending of any protest messages, as required by BFP [18, 107] and GDBF [187, 188], discussed in Section 4.5.

Algorithm rPDT can be applied in any application that requires on demand construction of a node's local view on planar graphs. Most notably these are applications in geographic unicast and multicast routing. For a comprehensive list of applications see the list provided in the motivation (Section 1.1). In addition, it can be incorporated in those applications that make explicit use of PDT, such as [53, 55, 56]. Therein, PDT is computed by a node based on full one-hop neighborhood information. Algorithm rPDT can be used to improve these algorithms' message efficiency.

Chapter 7

Reactive Geographic Routing on Euclidean Unit Disk Graph Spanners

In this chapter, the implications of PDT being a constant stretch Euclidean spanner of the unit disk graph on the state-of-the-art algorithms in beaconless Recovery routing are presented.

Except for Rotational Sweep (RS) [15, 68, 69] and ARROW [212], all of the beaconless recovery algorithms that guarantee message delivery in combination with Greedy routing (see survey in Section 4.3), make use of Gabriel graph planarization. Gabriel graphs constructed over unit disk graphs have Euclidean spanning ratios of $\Theta(\sqrt{n})$. In consequence, using these edges for recovery can lead to severe detours, which negatively affects the resource efficiency of the routing operation.

Although for ARROW [212] it is unknown if the edges used for recovery possibly impose a large stretch (in the worst-case), it is deficient regarding message complexity. Selection of the next routing hop requires $\Theta(n)$ message transmissions.

By far the best beaconless Recovery routing algorithm currently known is Rotational Sweep (RS) [15, 68, 69]. Using this algorithm, forwarding of the data packet along a guaranteed delivery recovery path requires only three message transmissions. RS can be combined with two delay functions, namely, *Sweep Circle* (SC) and *Twisting Triangle* (TT). If using RS with TT (RS-TT), the edges used for recovery are guaranteed to belong to a supergraph of the relative neighborhood graph. If instead using RS with SC (RS-SC), then the recovery edges even belong to a supergraph of the Gabriel graph. Hence, in particular for RS-SC it can be reasoned that the routing paths are potentially shorter regarding Euclidean distance than those based on actual Gabriel graphs. However, for now, it is unknown whether this supergraph of the Gabriel graph provides a constant Euclidean spanning ratio, or not.

Obviously, it would be beneficial if it can be guaranteed that a routing path only uses edges of a constant stretch spanner. Although local routing decisions may always fail to route a message along the globally shortest path, local routing in a constant stretch spanner potentially yields much shorter routing paths than routing along edges of a subgraph with non-constant spanning properties.

The first contribution of this chapter, presented in Section 7.2, is that RS-SC always uses edges that belong to PDT. In fact, it is shown that Recovery routing using FACE routing [36] on PDT and RS-SC produce equivalent routing paths. Hence, edges used by

RS-SC do not just belong to a supergraph of the Gabriel Graph, but belong to a constant stretch Euclidean spanner.

The second contribution of this chapter, given in Section 7.3, is an analysis of properties of paths produced by RS in general. It is shown that both RS-SC and RS-TT can produce arbitrarily bad routing paths in terms of number of hops. However, sometimes paths produced by RS-SC are beneficial compared to those produced by RS-TT, and vice versa.[1] These considerations then lead to another contribution.

The third contribution, presented in Section 7.4, is a guaranteed delivery reactive recovery algorithm, called *RS-Shortcut*. Essentially, it combines and extends the advantages of RS-SC and RS-TT. Routing paths produced by this algorithm are at most as long as those produced by RS-SC and may help to skip arbitrarily many unnecessary hops, while requiring at most eight message transmissions per forwarding step.

Prior to the presentation of these results, the underlying network model and assumptions are introduced in Section 7.1. The chapter concludes in Section 7.5 with a discussion and further implications.

7.1 Preliminaries

In the remainder, network graphs are modeled as unit disk graphs $\mathrm{UDG}(V)$, where $V \subset \mathbb{R}^2$ is a finite and distinct node set of size $|V| = n$. The unit disk radius is denoted by R. Henceforth, it is assumed that $\mathrm{UDG}(V)$ is connected, which is necessary to ensure that packets can successfully be routed between arbitrary node pairs. Moreover, as in preceding chapters, and for the same reasons, it is assumed that V neither contains any three collinear points, nor any four cocircular points (cf. Assumptions 2.1 & 2.2).

Given two non-adjacent nodes v and d from $\mathrm{UDG}(V)$, node v is called *local minimum node* (or simply *local minimum*) w.r.t. d, if no node in $\mathrm{N}_1(v)$ is closer to d than v itself (see Definition 2.7).

Definition 7.1 (Face traversal/recovery path)**.** Let $\mathrm{UDG}(V)$ be a connected unit disk graph and H a connected and planar subgraph of $\mathrm{UDG}(V)$. Let $v, d \in V$ s.t. v is local minimum w.r.t. d. The repeated application of FACE routing [36] (using the left hand rule) on the planar subgraph H starting at v with destination d yields a node sequence $\langle v = v_1, v_2, ..., v_k = d \rangle$, such that $v_i v_{i+1}$, $1 \leq i < k$, is an edge in $\mathrm{UDG}(V)$. This node sequence is called a *H-Face traversal*. Let v_j be the first node in this sequence for which $\|v_j d\| < \|vd\|$ holds. The *H-Face recovery path* of v w.r.t. destination d is given by the node sequence $S_H = \langle v = v_1, v_2, ..., v_j \rangle$.

Definition 7.2 (Rotational Sweep traversal)**.** Let $\mathrm{UDG}(V)$ be a unit disk graph and $v, d \in V$ s.t. v is local minimum w.r.t. d. The node sequences S_{SC} and S_{TT} that result from applying algorithm Rotational Sweep on v using the Sweep Circle (SC) and Twisting Triangle (TT) delay functions, respectively, are referred to as *SC-* and *TT-traversals*. The subsequences of S_{SC} and S_{TT}, starting at v and ending at the first node whose distance

[1] This was previously shown by Rührup et al. [15].

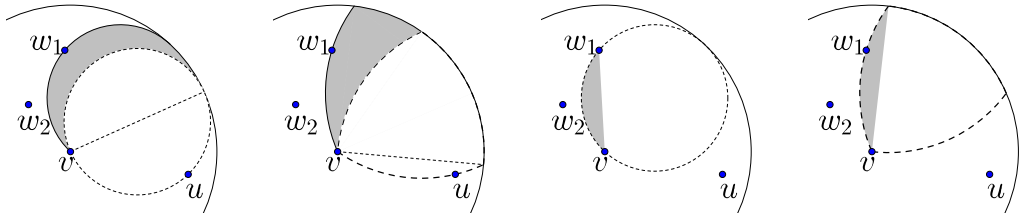


Figure 7.1: (a) SC sweep area of v w.r.t. u and w_1. (b) TT sweep area of v w.r.t. u and w_1. (c) $\mathrm{SC}(v, w_1)$ with $\mathrm{FR}^{\mathrm{SC}}(v, w_1)$ being shaded. (d) $\mathrm{TT}(v, w_1)$ with $\mathrm{FR}^{\mathrm{TT}}(v, w_1)$ being shaded.

to d is strictly smaller than $\|vd\|$ are then called the *SC-* and *TT-recovery paths* of v w.r.t. d.

The *sweep circle* and *twisting triangle* hinged at a node v and touching node w_1 are denoted by $\mathrm{SC}(v, w_1)$ and $\mathrm{TT}(v, w_1)$, respectively (see dashed circle and Releaux triangle in Figures 7.1c & 7.1d).

Definition 7.3 (Sweep area)**.** The *sweep area* of node v w.r.t. previous-hop u and next-hop w_1, is the area touched by the sweep curve, when rotated from the initial position (touching u) until it hits w_1 [15] (given by the shaded areas in Figures 7.1a & 7.1b).

By [15, Lemma 3] it holds that the sweep area is always empty of other nodes.

Definition 7.4 (Forbidden region)**.** Given two consecutive edges uv and vw_1 on a RS traversal, line segment $\overline{vw_1}$ divides $\mathrm{SC}(v, w_1)$ and $\mathrm{TT}(v, w_1)$ into two areas. The area containing points p with $\measuredangle uvp > \measuredangle uvw_1$ is called *forbidden region* [15] and is denoted by $\mathrm{FR}^{\mathrm{SC}}(v, w_1)$ and $\mathrm{FR}^{\mathrm{TT}}(v, w_1)$, respectively (given by the shaded areas in Figures 7.1c & 7.1d).

7.2 SC and PDT Traversals are Equivalent

In this section it is shown that PDT-Face and SC-traversals are equivalent if applied onto the same local minimum situation. This is captured by Theorem 7.8. Its proof requires few observations that are introduced first.

Fact 7.5. Let $A \subset \mathbb{R}^2$ be any area which is fully contained by the unit disk of node u. Then $A \cap \mathrm{N}_1(u) = A \cap V$.

Lemma 7.6. *Let $S_{\mathrm{SC}} = \langle v_0, ..., v_k \rangle$ be any sweep circle traversal. The area defined by the union of the sweep circle at v_0 in its initial position and the* SC *sweep areas of nodes v_i, $0 \leq i < k$, is empty of nodes from $V \setminus \{v_0, ..., v_k\}$.*

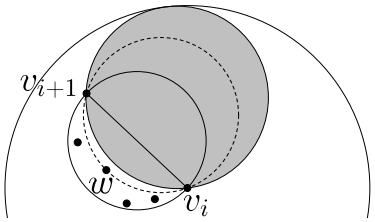


Figure 7.2: Illustration for the proof of Theorem 7.7. $\mathrm{C}(v_i, v_{i+1}, w)$ is dashed, $\mathrm{SC}(v_i, v_{i+1})$ is shaded, and $\mathrm{D}(v_i, v_{i+1})$ is solid.

Proof. The claim is proven by induction on index i.
I.h.: The claim holds for an arbitrary but fixed $i < k - 1$.
B.c. ($i = 0$): Node v_0 is a local minimum w.r.t. some destination node $d \in V$ and hence, $\mathrm{C}_{\|dv_0\|}(d)$ is empty of nodes from $\mathrm{N}_1(v_0) \setminus \{v_0\}$. The sweep circle at v_0 in initial position is entirely contained by $\mathrm{C}_{\|dv_0\|}(d)$ and hence, it is empty of nodes from $\mathrm{N}_1(v_0) \setminus \{v_0\}$. By Lemma 3 in [15] it holds that the sweep area of v_0 w.r.t. v_1 and the initial position is empty of nodes from $\mathrm{N}_1(v_0) \setminus \{v_0, v_1\}$. With Fact 7.5, the claim follows.
I.s. ($i \to i + 1$): By the i.h. it holds that the area defined by the union of the sweep circle at v_0 in its initial position and the SC sweep areas of nodes v_i, $0 \leq i < k - 1$, is empty of nodes from $V \setminus \{v_0, ..., v_{i+1}\}$. By Lemma 3 in [15] it holds that the sweep area of v_{i+1} w.r.t. v_i and v_{i+2} is empty of nodes from $\mathrm{N}_1(v_{i+1}) \setminus \{v_{i+1}, v_{i+2}\}$. With Fact 7.5, the claim follows. □

Theorem 7.7. *Let S_{SC} be a sweep circle traversal and let v_i and v_{i+1} be any pair of successive nodes in S_{SC}. Then, $v_i v_{i+1}$ is an edge in* $\mathrm{PDT}(V)$.

Proof. Let S_{GG} denote the Gabriel Face traversal corresponding to S_{SC}. Rührup and Stojmenović prove in Lemma 2 in [68] that $S_{\mathrm{SC}} \subseteq S_{\mathrm{GG}}$, i.e., each node contained in S_{SC} is also visited in S_{GG}. If v_i and v_{i+1} are successors in S_{GG}, then $v_i v_{i+1} \in \mathrm{GG}(V) \cap \mathrm{UDG}(V)$. Moreover, $\mathrm{GG}(V) \cap \mathrm{UDG}(V) \subseteq \mathrm{PDT}(V)$ holds by definition of PDT. Hence, in this case $v_i v_{i+1} \in \mathrm{PDT}(V)$ follows.

Otherwise, if v_i and v_{i+1} are not successors in S_{GG}, there exists at least one other node in $\mathrm{D}(v_i, v_{i+1})$. Let $w \in V$ be the angle maximizing node w.r.t. $v_i v_{i+1}$ (see Figure 7.2). Because v_i and v_{i+1} are successors in S_{SC}, w must be outside of $\mathrm{SC}(v_i, v_{i+1})$. This implies that the diameter of $\mathrm{C}(v_i, v_{i+1}, w)$ is strictly smaller than R, which is the diameter of $\mathrm{SC}(v_i, v_{i+1})$. Hence, it remains to show that $\mathrm{C}(v_i, v_{i+1}, w)$ is empty of nodes from $\mathrm{N}_1(v_i) \setminus \{v_i, v_{i+1}, w\}$. By Lemma 7.6 this holds for $\mathrm{C}(v_i, v_{i+1}, w) \cap \mathrm{SC}(v_i, v_{i+1})$. Because w is angle maximizing, this also holds for $\mathrm{C}(v_i, v_{i+1}, w) \setminus \mathrm{SC}(v_i, v_{i+1})$. □

Theorem 7.8. $S_{\mathrm{PDT}} = S_{\mathrm{SC}}$.

Proof. Let $S_{\mathrm{PDT}} = \langle v = p_0, p_1, ..., p_k \rangle$ denote the PDT Face traversal starting at local minimum node v, and let $S_{\mathrm{SC}} = \langle v = v_0, v_1, ..., v_l \rangle$ denote the corresponding SC traversal.

By induction on index i it is shown that (1) $S_{\mathrm{SC}} \subseteq S_{\mathrm{PDT}}$ and (2) $S_{\mathrm{PDT}} \subseteq S_{\mathrm{SC}}$.
I.h.: For an arbitrary but fixed i with

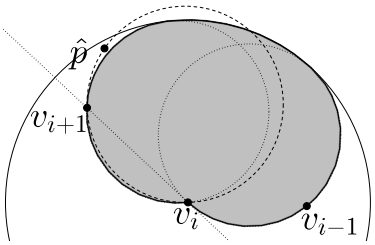


Figure 7.3: Illustrations for the proof of Theorem 7.8. The sweep area of v_i w.r.t. v_{i+1} and v_{i-1} is shaded, $\mathrm{C}(v_i, v_{i+1}, \hat{p})$ is dashed.

i) $i < l$, $v_i \in S_{\mathrm{PDT}}$

ii) $i < k$, $p_i \in S_{\mathrm{SC}}$

B.c. ($i = 0$): With $v_0 = v = p_0$ both claims hold.
I.s. ($i \to i + 1$): At first, claim (1) is shown.

By part i) of the i.h., for all $v_0, ..., v_i$ it holds that they are contained by S_{PDT}. By Lemma 7.6 the sweep area of v_i w.r.t. v_{i-1} and v_{i+1} is empty of nodes from $V \setminus \{v_0, ..., v_{i+1}\}$ and v_{i+1} is the first node to be hit by v_i's sweep circle.

Suppose $v_{i+1} \notin S_{\mathrm{PDT}}$, then the successor of v_i in S_{PDT}, say $\hat{p}$, must be located clockwise of $\overrightarrow{v_i v_{i+1}}$ and outside the sweep area of v_i (see Figure 7.3). This holds because edges along S_{SC} always belong to $\mathrm{PDT}(V)$ by Theorem 7.7 and edge $v_i\hat{p}$ is, by construction of S_{PDT}, the first counter-clockwise edge from $\mathrm{PDT}(V)$ w.r.t. previous-hop v_{i-1}.

For any possible position of $\hat{p}$, the maximal angle w.r.t. $v_i\hat{p}$, and a third node $x \in V \setminus \{v_i, \hat{p}\}$ is at least as large as angle $\measuredangle v_i v_{i+1}\hat{p}$. Because $\mathrm{SC}(v_i, v_{i+1})$ with diameter R does not touch $\hat{p}$, and $\hat{p}$ is clockwise of $\overrightarrow{v_i v_{i+1}}$, the diameter of $\mathrm{C}(v_i, v_{i+1}, \hat{p})$ is strictly larger than R. But this implies that $v_i\hat{p} \notin \mathrm{PDT}(V)$, which contradicts the assumption that $\hat{p}$ is the successor of v_i in S_{PDT}. Hence, $v_{i+1} \in S_{\mathrm{PDT}}$ must hold.

Now consider claim (2). By part ii) of the i.h., for all $p_0, ..., p_i$ it holds that they are contained by S_{SC}. Note that this and the fact that claim (1) has already been proven implies that $p_j = v_j$, for all $0 \leq j \leq i$.

Edge $p_i p_{i+1}$ is the first counter-clockwise PDT edge w.r.t. previous-hop p_{i-1}. There exists a circle C of diameter at most R, which has p_i, p_{i+1}, and at most one more node on its boundary, and which is empty of other nodes from V. Assume $p_{i+1} \notin S_{\mathrm{SC}}$. Then the sweep circle hinged at p_i hits v_{i+1} before it hits p_{i+1}.

If v_{i+1} is counter-clockwise of $\overrightarrow{p_i p_{i+1}}$, then v_{i+1} must be located in $\mathrm{D}(p_i, p_{i+1})$ and for the angle maximizing node w w.r.t. $p_i p_{i+1}$ it then holds that $\measuredangle p_i w p_{i+1} \geq \measuredangle p_i v_{i+1} p_{i+1}$. Since $\mathrm{SC}(p_i, v_{i+1})$ has diameter R and it does not contain p_{i+1}, it follows that the diameter of $\mathrm{C}(p_i, p_{i+1}, w)$ must be strictly larger than R. But this is a contradiction to $p_i p_{i+1} \in \mathrm{PDT}(V)$.

In the remaining case, v_{i+1} is in clockwise direction of $\overrightarrow{p_i p_{i+1}}$. By Theorem 7.7, $p_i v_{i+1} \in \mathrm{PDT}(V)$ holds. But this contradicts the assumption that p_{i+1} succeeds p_i in S_{PDT}.

It can be concluded that $p_{i+1} \in S_{\mathrm{SC}}$ holds, which proves the claim. □

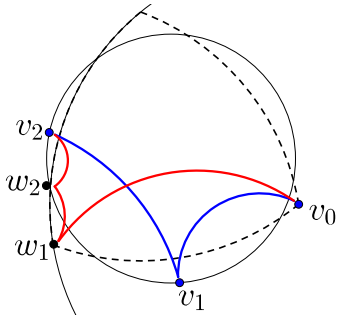


Figure 7.4: The TT traversal $\langle v_0, w_1, w_2, v_2 \rangle$ (red) and the SC traversal $\langle v_0, v_1, v_2 \rangle$ (blue) are partially disjoint. This Figure is similar to Figure 22 in [15].

7.3 Path Properties of Rotational Sweep Traversals

The results obtained in the previous section are now combined with previous results on RS traversals and their relations to planar subgraph traversals. Ultimately, these considerations lead to a scheme describing how to cut short in SC traversals without violating the delivery guarantee.

The combination of Theorem 2 in [69], Lemma 2 in [68], and Theorem 7.8 yields the following set of subpath relations, where $S \supseteq S'$, if each node in S' is also in S, whereas $S \not\supseteq S'$, if there exist scenarios such that a node in S' is not contained in S.[2]

1. $S_{\text{RNG}} \supseteq S_{\text{GG}} \supseteq S_{\text{SC}} = S_{\text{PDT}}$
2. $S_{\text{GG}} \not\supseteq S_{\text{TT}}$ and $S_{\text{SC}} \not\supseteq S_{\text{TT}}$
3. $S_{\text{RNG}} \supseteq S_{\text{TT}}$

Figure 7.4 shows that TT and SC traversals may be partially disjoint (this example has previously been given in [15, 69]). This implies in particular that TT traversals make use of non-PDT edges. For example, edge v_0w_1 in Figure 7.4 is not a PDT edge, but it used by the TT traversal. Hence, the property of SC traversals proved in the previous section, namely that SC traversals belong to PDT which is a constant stretch Euclidean spanner of the unit disk graph, does not hold in general for TT traversals.

However, Rührup and Stojmenović [15] give empirical evidence that TT traversals yield shorter paths, regarding Euclidean path length, compared to traversals produced by SC or Angular Relaying (AR) [18, 107] (see Section 4.3). To the best of the author's knowledge, no theoretical bounds for TT traversals are known, apart from those listed above. In fact there are node constellations where TT traversals produce detours compared to the corresponding SC traversals. E.g., in Figure 7.4 the TT traversal is 6% longer than the SC traversal, regarding Euclidean length. This example demonstrates in addition that TT traversals may also exceed SC traversals regarding hop count.

[2] Note that different notations for subpath relations have previously been used; compare e.g. [68] and [69]. Here, the notation from [68] is used, which is explained above.

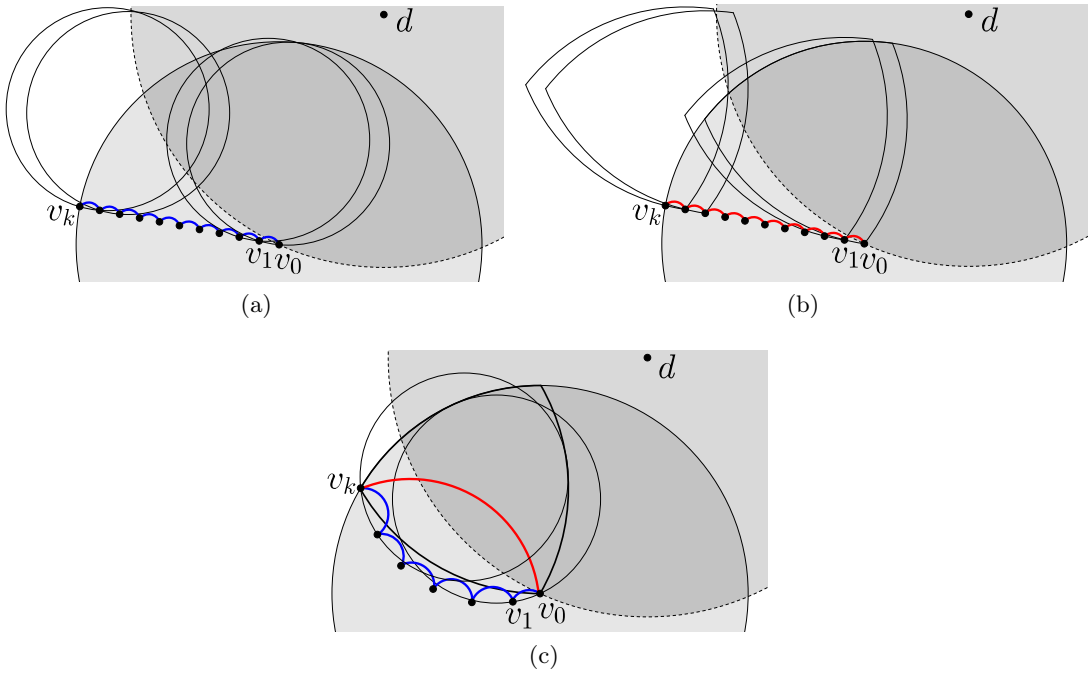


Figure 7.5: (a) and (b) are examples where RS using SC (blue) and TT (red), respectively, performs particularly bad and requires $\Theta(n)$ hops within unit transmission range R; (c) while the SC traversal $\langle v_0, v_1, ..., v_k \rangle$ makes $\Theta(n)$ hops, the TT traversal $\langle v_0, v_k \rangle$ skips these unnecessary hops.

Looking at the hop count, there are examples where RS traversals in general perform arbitrarily bad. Consider the node constellations depicted in Figures 7.5a and 7.5b, where v_0 is a local minimum node w.r.t. destination node d and where nodes $\{v_0, v_1, ..., v_k\}$ are aligned on a line segment of unit length R.[3] Independent of whether SC or TT delay function is used, the RS algorithm outputs the sequence $\langle v_0, v_1, ..., v_k, ... \rangle$. Obviously, if k is in the order of the number of nodes n, such traversals make $\Theta(n)$ hops, although there is a direct communication link between v_0 and v_k in the underlying unit disk graph.

In contrast, there are examples where a single TT traversal edge bypasses an SC traversal consisting of $\Theta(n)$ hops. For an example consider Figure 7.5c, where the TT edge v_0v_k skips the multi-hop SC traversal $\langle v_0, v_1, ..., v_k \rangle$.

This example, combined with the above listed facts on RS traversals, suggests the following algorithmic strategy for obtaining short recovery paths: "Take the next TT edge instead of the next SC edge, whenever it helps to cut short at least two hops on the SC traversal". This strategy, however, leads to paths that are not necessarily subpaths of SC traversals (e.g., v_k in Figure 7.5c could be a node like w_1 in Figure 7.4, leading to

[3] It is easy to construct similar examples without aligning nodes along a straight line and hence, assuming nodes not to be collinear does not resolve this problem.

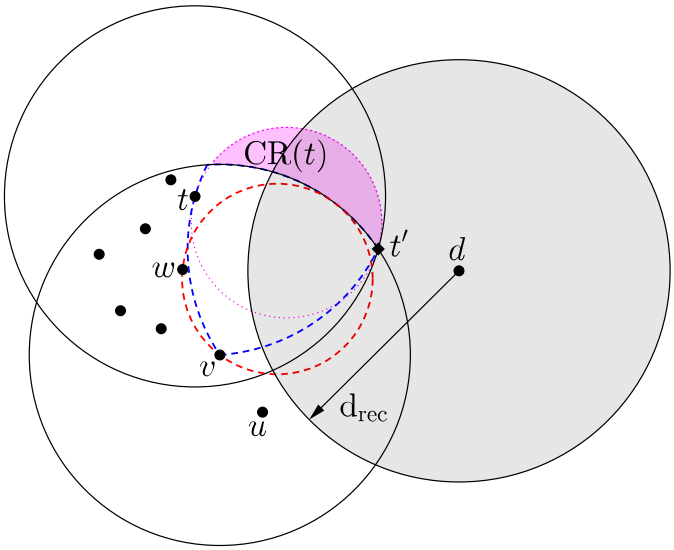


Figure 7.6: Illustration of the idea underlying algorithm RS-Shortcut, where u is the previous-hop, v is the current forwarder and w and t are the next hops on a SC and TT traversal, respectively. The pink colored area represents the critical region incident to t.

an increase in hop or Euclidean path length). Most important, the properties proven in Section 7.2 for SC traversals do not apply to such paths.

Therefore, the beaconless algorithm introduced next chooses to cut short on a SC path by taking a TT edge, if and only if the shortcut leads to a node, which is guaranteed to be part of the corresponding SC traversal. This way, the properties of SC traversals are maintained, while per hop up to $\Theta(n)$ unnecessary routing steps can be avoided.

7.4 Beaconless Bypassing of Sweep Circle Detours

The following beaconless recovery algorithm, called *RS-Shortcut*, is based on the observations from the previous section. It can be considered to be an extension of the Rotational Sweep algorithm [15]. Its purpose is to construct a recovery path along which a data packet is being forwarded, starting at a local minimum node and ending at a node whose distance to the packet's destination is strictly smaller compared to the *recovery distance* $\mathrm{d_{rec}}$, the distance between the local minimum and the destination.

As opposed to the original RS algorithm, where the data packet is always directly forwarded to the next counter-clockwise SC and TT neighbor, respectively, algorithm RS-Shortcut adapts dynamically to whichever choice is beneficial. It tries to make as much progress as possible, provided that the next-hop is guaranteed to be a node on the SC recovery path. This is achieved with the aid of (very limited) partial two-hop neighborhood information. For the following illustration of the main idea, see Figure 7.6.

A node v in a recovery situation determines first its counter-clockwise SC and TT neighbors w and t, respectively, by means of algorithm RS. If these nodes coincide, this node is used for forwarding. Otherwise, the TT neighbor t determines reactively means of a contention mechanism, whether there is at least one node in the *critical region* ($\mathrm{CR}(t)$) incident to it, or not. If the critical region happens to be empty, the TT neighbor t is selected for forwarding, otherwise the SC neighbor w. The critical region, given by the pink colored area in Figure 7.6, is chosen such that t can verify reactively, using a single contention, if it is empty or not. It is defined as follows.

Definition 7.9 (Critical region)**.** Given a twisting triangle $\mathrm{TT}(v,t)$, let t' be the corner of the twisting triangle that is not incident to the arc connecting v and t. The *critical region incident to* t, denoted $\mathrm{CR}(t)$, is given by the part of the closed disk $\mathrm{D}(t,t')$ that is outside of the unit disk of v. Formally,

$$\mathrm{CR}(t) = \mathrm{D}(t,t') \setminus \mathrm{C_R}(v)\,.$$

Since the diameter of a twisting triangle is equal to the unit disk radius R, it holds that $\mathrm{D}(t,t')$ has diameter R. Hence, $\mathrm{CR}(t)$ is contained by the unit disk centered at t. If it contains a node, than this node is a neighbor of t in the underlying unit disk graph.

7.4.1 Algorithm RS-Shortcut

Algorithm RS-Shortcut is executed by a node v, if and only if it is holding a DATA packet in *Recovery mode*. That is, v is either a local minimum node w.r.t. some destination d, or it is a node on a recovery path being at least as far away from d than recovery distance $\mathrm{d_{rec}} = \|v_0 d\|$, the Euclidean distance between the local minimum v_0 at which the recovery process has started and d.

In the following description, u denotes the previous-hop on the recovery path from which executing node v has received the DATA packet. Its position is required for computation of the delay functions $\mathrm{t_{SC}}(\cdot,\cdot)$ and $\mathrm{t_{TT}}(\cdot,\cdot)$, which refer to the sweep circle and twisting triangle delay functions described in [15] (see also Figure 4.8). In case there is no such previous-hop, the intersection of $\overline{vd}$ with the unit disk centered at v is used instead (cf. Section 4.3). $\mathrm{t_{max}} > 0$ is a positive constant that is either known to all nodes or is part of the packet header.

Description of execution of RS-Shortcut by node v:

Node v locally broadcasts $\mathrm{RTS}(\mathrm{SC}, v, u)$ and waits for the first $\mathrm{CTS}(\mathrm{SC}, w, v)$ response by a neighbor $w \in \mathrm{N_1}(v)$. If w is closer to the destination than the recovery distance, then v switches the DATA packet to *Greedy mode*, forwards it immediately to w, and terminates.

Otherwise, v locally broadcasts $\mathrm{RTS}(\mathrm{TT}, v, u)$ and waits until it receives the first $\mathrm{CTS}(\mathrm{TT}, t, v)$ of a neighbor $t \in \mathrm{N_1}(v)$.

If the SC and TT neighbors w and t coincide or TT neighbor t is closer to the destination than the recovery distance, then v immediately forwards the DATA packet to w and terminates.[4]

Otherwise, if $w \neq t$, then v checks if w and d are in the same half-plane w.r.t. $\ell(v,t)$ and if $\mathrm{FR}^{\mathrm{SC}}(w,t) \cap \mathrm{C}_{\mathrm{d_{rec}}}(d) \neq \emptyset$ (see Figure 7.10). If this is the case, then routing to t potentially leaves the SC recovery paths and therefore v forwards the DATA packet immediately to w. In the other case, v needs to know if the critical region incident to t is empty of nodes. It triggers this validation by sending $\mathrm{RTS}(\mathrm{SCut}, v, t)$ and waits until it receives a $\mathrm{CTS}(\mathrm{SCut}, t, v, b)$ from t. If $b = 1$ (i.e., the critical region incident to t is empty of nodes), then v forwards the DATA packet to t. Otherwise, if $b = 0$ (i.e., the critical region is not empty), then v forwards the DATA packet to w. In either case, node v terminates after sending the DATA packet.

On reception of the $\mathrm{RTS}(\mathrm{SCut}, v, t)$ by TT neighbor t of v, node t computes the critical region $\mathrm{CR}(t)$ incident to it, which is defined by the positions of v, t and distinguished point t'. Then, it locally broadcasts $\mathrm{RTS}(\mathrm{CR}, t, v, t')$ and starts a delay timer of duration $\mathrm{t_{max}}$. Those neighbors x of t that are located in $\mathrm{CR}(t)$ overhear this message and schedule sending of a $\mathrm{CTS}(\mathrm{CR}, x, t)$ after a delay of duration $(\|xt\|/\mathrm{R}) \cdot \mathrm{t_{max}}$. Now two cases can occur. Either there is a neighbor x of t in the critical region, then this neighbor responds with a message prior to expiration of t's delay timer. In this case, t immediately sends $\mathrm{CTS}(\mathrm{SCut}, t, v, 0)$ to v. Otherwise, if there is no such neighbor x, then after time $\mathrm{t_{max}}$ its delay timer expires and it can be sure that there is no neighbor in the critical region. Then, it sends $\mathrm{CTS}(\mathrm{SCut}, t, v, 1)$ to v.

Generally, upon overhearing CTS responses, new RTS requests, or transmissions of the DATA packet, nodes that have neither initiated algorithm execution nor are the corresponding SC and TT neighbors cancel their running delay timers and sending of scheduled messages. This ensures that an RTS generates at most one CTS answer.

A detailed pseudocode description of algorithm RS-Shortcut is given by Algorithm 2.

7.4.2 Correctness and Analysis

In the following proofs, v always refers to the node executing the algorithm, whereas w and t always refer to the SC and TT neighbor of v, respectively.

Theorem 7.10 (Message complexity and size)**.** *During execution of RS-Shortcut at most nine messages are transmitted. The size of* RTS/CTS *messages is at most* $\mathcal{O}(\mathrm{P_{max}})$ *bits, where* $\mathrm{P_{max}}$ *denotes the number of bits required to represent a single geographic position.*

Proof. Observe that any RTS sent during execution of RS-Shortcut results in a single CTS answer for the following reason. There are only two nodes that broadcast RTS messages, namely, forwarder v and possibly its TT neighbor t. Upon reception of the corresponding CTS both v and t always respond immediately either with another RTS or DATA, which cancel running timers of their neighbors and suppress sending of further CTS messages.

[4] Forwarding to the SC neighbor w instead of the TT neighbor t simply ensures that the DATA packet is forwarded only to nodes that actually belong to the corresponding SC traversal.

Algorithm 2 RS-Shortcut

Variables: u is the previous-hop, $\mathrm{d_{rec}}$ is the recovery distance, d is the destination of the DATA packet, R is the unit disk transmission radius, and $\mathrm{t_{max}} > 0$ is a constant known to all nodes.

Definition of message types

RTS(SC, v, u**)** request by v to start the SC delay timer w.r.t. v and previous-hop u

CTS(SC, w, v**)** response by w to the corresponding $\mathrm{RTS}(\mathrm{SC}, v, u)$

RTS(TT, v, u**)** request by v to start the TT delay timer w.r.t. v and previous-hop u

CTS(TT, t, v**)** response by t to the corresponding $\mathrm{RTS}(\mathrm{TT}, v, u)$

RTS(SCut, v, n_{tt}**)** request by v destined at its TT neighbor n_{tt} to verify if the critical region incident to it is empty, or not

CTS($\mathrm{SCut}, n_{tt}, v, b$**)** response by TT neighbor n_{tt} of v to the corresponding $\mathrm{RTS}(\mathrm{SCut}, v, n_{tt})$ containing boolean variable b, where $b = 1$ if the critical region of n_{tt} is empty and $b = 0$, otherwise

RTS($\mathrm{CR}, n_{tt}, v, t'$**)** request of v's TT neighbor n_{tt}, if the critical region defined by positions, n_{tt}, t', and v, is empty of nodes, or not

CTS(CR, x, n_{tt}**)** response by x to the corresponding $\mathrm{RTS}(\mathrm{CR}, n_{tt}, v, t')$ serving as a witness that the critical region of n_{tt} is not empty

Action by initiator v

Initialize local variables n_{sc} and n_{tt} representing the next SC and TT neighbor of v
Locally broadcast $\mathrm{RTS}(\mathrm{SC}, v, u)$ and wait for reception of the corresponding response
on reception of $\mathrm{CTS}(\mathrm{SC}, w, v)$ **do**
    $n_{sc} \leftarrow w$
    if $\|dn_{sc}\| < \mathrm{d_{rec}}$ **then**
        Switch DATA packet to *Greedy mode*, forward it immediately to n_{sc}, and terminate
    else
        Locally broadcast $\mathrm{RTS}(\mathrm{TT}, v, u)$ and wait for reception of the corresponding response
on reception of $\mathrm{CTS}(\mathrm{TT}, t, v)$ **do**
    $n_{tt} \leftarrow t$
    if $n_{sc} = n_{tt}$ or $\|dn_{tt}\| < \mathrm{d_{rec}}$ **then**
        Immediately forward DATA packet to n_{sc} and terminate
    else ▷ Note that $n_{sc} \neq n_{tt}$ and $\|dn_{tt}\| \geq \mathrm{d_{rec}}$ holds
        if n_{sc} and d are in the same half-plane w.r.t. $\ell(v, n_{tt})$ and $\mathrm{FR}^{\mathrm{SC}}(n_{sc}, n_{tt}) \cap \mathrm{C}_{\mathrm{d_{rec}}}(d) \neq \emptyset$ **then**
            Forward the DATA packet immediately to n_{sc} and terminate
        else
            Locally broadcast $\mathrm{RTS}(\mathrm{SCut}, v, n_{tt})$ and wait for reception of the corresponding response
on reception of $\mathrm{CTS}(\mathrm{SCut}, n_{tt}, v, b)$ **do**
    if $b = 1$ **then** ▷ Critical region incident to n_{tt} is empty of nodes
        Forward DATA packet to n_{tt}
    else ▷ Critical region incident to n_{tt} is not empty
        Forward DATA packet to n_{sc}
    Terminate

(Continued on next page)

Algorithm 2 (continuation)

Action by any node $z \neq v$

on reception of RTS(SC, v, u) **do**
    Compute $\Theta_z = \measuredangle uvz$ and schedule sending of CTS(SC, z, v) after delay of duration $t_{SC}(\|vz\|, \Theta_z)$
on reception of CTS(SC, y, v) **do**
    Cancel delay timer and sending of CTS(SC, z, v) (if any)
on reception of RTS(TT, v, u) **do**
    Cancel running delay timer and sending of CTS(SC, z, v) (if any), compute $\Theta_z = \measuredangle uvz$, and schedule sending of CTS(TT, z, v) after delay of duration $t_{TT}(\|vz\|, \Theta_z)$.
on reception of CTS(TT, y, v) **do**
    Cancel delay timer and sending of CTS(TT, z, v) (if any)
on reception of RTS(SCut, v, n_{tt}) **do**
    if $z \neq n_{tt}$ **then**
        Cancel running relay timer and sending of CTS(TT, z, v) (if any)
    else ▷ $z = n_{tt}$, i.e.,z is TT neighbor of v
        Compute CR(n_{tt}), locally broadcast RTS(CR, n_{tt}, v, t'), where t' is the distinguished point defining CR(n_{tt}), and start a delay timer of duration t_{max}. If within this delay period no CTS(CR, x, n_{tt}) is received from a neighbor x, then locally broadcast CTS(SCut, $n_{tt}, v, 1$)
on reception of CTS(CR, x, n_{tt}) **do**
    if $z = n_{tt}$ **then**
        Cancel running delay timer and immediately send CTS(SCut, $n_{tt}, v, 0$)
    else
        Cancel delay timer and sending of CTS(CR, z, n_{tt}) (if any)
on reception of RTS(CR, n_{tt}, v, t') **do**
    if $z \in D(n_{tt}, t')$ and $\|zv\| > R$ **then** ▷ I.e., if z is in the critical region incident to n_{tt}
        Schedule sending of CTS(CR, z, n_{tt}) after delay of duration $t(\|n_{tt}z\|) = (\|n_{tt}z\|/R) \cdot t_{max}$
on reception of DATA **do**
    Cancel all running delay timers and sending of messages (if any)
    if z is addressed in the packet header as next hop **then**
        Handle the packet according to the packet's *mode*, i.e., apply Greedy routing if the packet is in *Greedy mode*, and RS-Shortcut if the packet is in *Recovery mode*

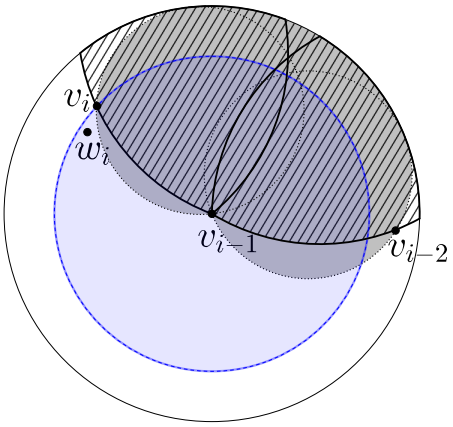


Figure 7.7: Illustration for the proof of Lemma 7.11. The TT sweep area is striped, whereas the SC sweep area is given by the gray shaded area. Circle $C_{\|v_{i-1}v_i\|}(v_{i-1})$ is represented by the blue area.

If algorithm execution terminates at line 6, exactly three messages have been transmitted. If instead it terminates at line 12 or 15, then five messages have been transmitted. Finally, if it terminates at line 23, then four messages have been transmitted for detection of the SC and TT neighbors w and t. Forwarder v as well as its TT neighbor t each broadcast another RTS. At most two CTS responses are generated thereupon. Finally, the DATA is forwarded and hence, at most nine messages are being transmitted.

Any RTS/CTS message contains at most a constant number of geographic positions plus possibly a constant number of additional bits. Thus, the size of these messages is $\mathcal{O}(P_{max})$ bits. □

The above proof relies on the following two assumptions. Firstly, no four nodes in V are allowed to be cocircular. Otherwise, the SC and TT delay functions yield simultaneous CTS replies which may lead to a collision. Secondly, no two neighbors in the critical region have the same distance to the TT neighbor t. Otherwise, a collision at t may occur. This is unproblematic if t is able to observe collisions, since a collision implies the existence of nodes in that area. Collisions due to simultaneous timeouts are a general problem of position-based delay functions and are discussed in detail in Section 10.2.

Next, the correctness of algorithm RS-Shortcut is proven, which requires the following auxiliary lemma.

Lemma 7.11. *Let $S_{SC} = \langle v = v_0, ..., v_l \rangle$ and $S_{TT} = \langle v = w_0, ..., w_k \rangle$ be the* SC *and* TT *traversals, respectively, starting at a particular local minimum node v, such that these traversals differ by at least one node. Let $0 < i \leq l$ be any index s.t. $v_{i-1} = w_{i-1}$ and $v_i \neq w_i$, i.e., the edges $v_{i-1}v_i$ and $w_{i-1}w_i$ differ. Then, $\|v_{i-1}v_i\| < \|w_{i-1}w_i\|$.*

Proof. For the sake of contradiction, assume $\|v_{i-1}v_i\| \geq \|w_{i-1}w_i\|$.

In case $\|v_{i-1}v_i\| = \|w_{i-1}w_i\|$, both v_i and w_i are on the boundary of the circle centered at $v_{i-1} = w_{i-1}$ with radius $\|v_{i-1}v_i\| = \|w_{i-1}w_i\|$ (blue circle in Figure 7.7). Because

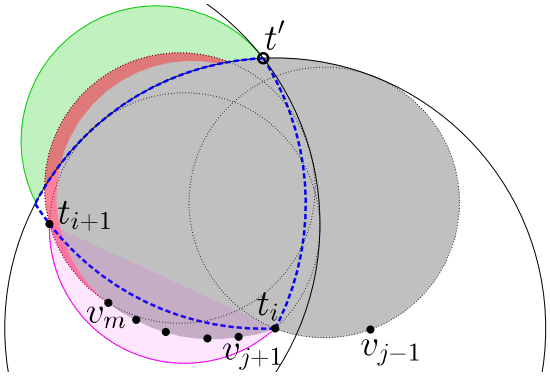


Figure 7.8: Illustration for the proof of Theorem 7.12. Sweep region produced by $\langle v_{j-1}, v_j = t_i, v_{j+1}, ..., v_m \rangle$ is given by the gray shaded area. $\mathrm{TT}(t_i, t_{i+1})$ is the blue dashed Releaux triangle. The pink area represents $\mathrm{FR}^{\mathrm{SC}}(t_i, t_{i+1})$, and the red area represents possible positions of v_m's SC traversal successor v_{m+1}. The green area represents $\mathrm{CR}(t_{i+1})$. The black circles are the unit disks of t_i and t_{i+1}.

$v_{i-1}v_i \neq w_{i-1}w_i$ either v_i, or w_i is the first node to be hit by both rotating sweep curves. But this is a contradiction to $v_i \neq w_i$ and hence, this case cannot occur.

In the other case, where $\|v_{i-1}v_i\| > \|w_{i-1}w_i\|$, w_i must be inside circle $\mathrm{C}_{\|v_{i-1}v_i\|}(v_{i-1})$, as illustrated in Figure 7.7.

Observe, that the union of $\mathrm{SC}(v_{i-1}, v_{i-2})$ and the sweep area of the sweep circle of v_{i-1} w.r.t. v_{i-2} and v_i (given by the gray shaded area in Figure 7.7), restricted on $\mathrm{C}_{\|v_{i-1}v_i\|}(v_{i-1})$, covers entirely the union of $\mathrm{TT}(v_{i-1}, v_{i-2})$ and the sweep area of the sweep triangle of v_{i-1} w.r.t. v_{i-2} and v_i (given by the striped area in Figure 7.7), also restricted on $\mathrm{C}_{\|v_{i-1}v_i\|}(v_{i-1})$. That is, if w_i is hit by the sweep triangle before the triangle hits v_i, then w_i is also hit first by the sweep circle. But this is a contradiction to $v_i \neq w_i$ and hence, this case can also not occur. Note that this argument also holds in the special case where $i = 1$ (i.e., the first edges of S_{SC} and S_{TT} differ) due to the sweep object's initial positions. □

Correctness of algorithm RS-Shortcut is proven by showing that the DATA packet is only forwarded to nodes belonging to the corresponding SC recovery path, which is known to be a guaranteed delivery recovery path.

Theorem 7.12. *Let S_{SCut} denote the node sequence produced by the RS-Shortcut algorithm if applied onto a local minimum node and let S_{SC} be the corresponding* SC *recovery path. It holds that $S_{\mathrm{SCut}} \subseteq S_{\mathrm{SC}}$.*

Proof. Let v be a local minimum node w.r.t. destination d. Let $S_{\mathrm{SCut}} = \langle v = t_0, t_1, ..., t_k \rangle$ be the path resulting from application of RS-Shortcut onto v and $S_{\mathrm{SC}} = \langle v = v_0, v_1, ..., v_l \rangle$ the corresponding SC recovery path. By induction on index i it is now proven that $t_i \in S_{\mathrm{SC}}$, for all $0 \leq i \leq k$.
I.h.: For an arbitrary but fixed $i < k$ it holds that $t_i \in S_{\mathrm{SC}}$.

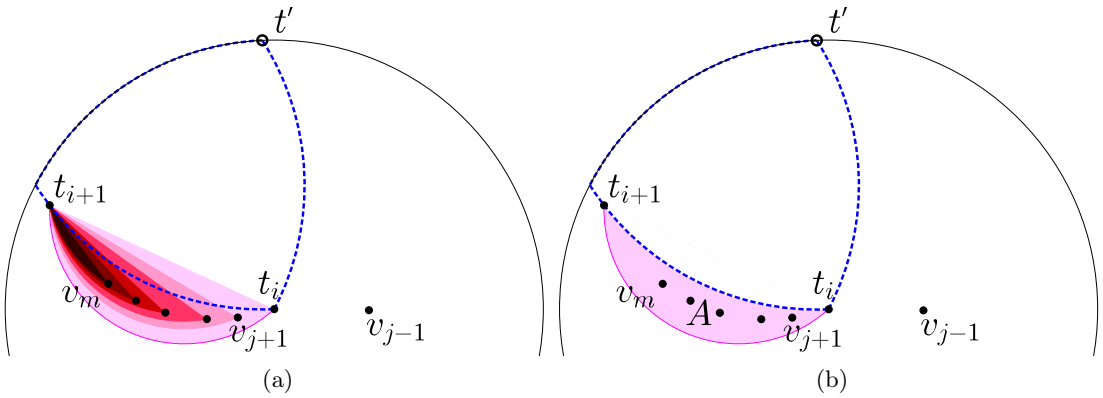


Figure 7.9: Illustration of (a) Equation 7.1 and (b) Equation 7.2.

B.c. ($i = 0$): With $t_0 = v = v_0$, $t_0 \in S_{\mathrm{SC}}$ holds trivially.
I.s. ($i \to i+1$): By the i.h., for all $i' \leq i$ it holds that $t_{i'} \in S_{\mathrm{SC}}$ and in particular $t_i = v_j$, for some $j \geq i$.

The fact that $i < k$ implies that $\|t_i d\| \geq \mathrm{d_{rec}}$ since RS-Shortcut is executed by a node, only if it is a local minimum or another node with distance at least $\mathrm{d_{rec}}$ from the destination. Since by Lemma 4 in [68], S_{SC} always terminates at a node with distance to the destination smaller than $\mathrm{d_{rec}}$, the existence of node $t_{i+1} \in S_{\mathrm{SCut}}$ implies the existence of node $v_{j+1} \in S_{\mathrm{SC}}$.

If $t_{i+1} = v_{j+1}$ the claim holds. Therefore, in the reminder of this proof it is assumed that $t_{i+1} \neq v_{j+1}$. The latter implies that t_{i+1} is the successor of t_i on a TT traversal (see Figure 7.8). Thus, Lemma 7.11 can be applied and yields that $\|t_i v_{j+1}\| < \|t_i t_{i+1}\|$. Moreover, due to the counter-clockwise sweeping direction, it holds that v_{j+1} is counter-clockwise of the directed edge $\overrightarrow{t_i t_{i+1}}$.

Let $\Sigma = \langle v_{j+1}, ..., v_m \rangle \subset S_{\mathrm{SC}}$ be the longest consecutive subsequence of S_{SC} which starts at v_{j+1} and such that all nodes of Σ are located in the same closed half-plane as v_{j+1} w.r.t. the straight line $\ell(t_i t_{i+1})$ (see nodes in the pink area in Figure 7.8).

If $t_{i+1} \in \Sigma$, then $t_{i+1} \in S_{\mathrm{SC}}$ holds. Therefore, consider the case $t_{i+1} \notin \Sigma$. Because $t_i \oplus \Sigma = \langle t_i, v_{j+1}, ..., v_m \rangle$ is a consecutive subsequence of S_{SC} not containing t_{i+1}, the following holds (see Figure 7.9a):

$$\mathrm{FR}^{\mathrm{SC}}(t_i, t_{i+1}) \supset \mathrm{FR}^{\mathrm{SC}}(v_{j+1}, t_{i+1}) \supset \cdots \supset \mathrm{FR}^{\mathrm{SC}}(v_m, t_{i+1})\,. \tag{7.1}$$

This implies in particular that all nodes from Σ are contained by $\mathrm{FR}^{\mathrm{SC}}(t_i, t_{i+1})$. Moreover, the nodes in Σ must be outside of $\mathrm{FR}^{\mathrm{TT}}(t_i, t_{i+1})$, since the sweep area produced by a sweep triangle is empty of nodes according to Lemma 3 in [15]. Hence, for all nodes in Σ it holds that they are located in area A (see Figure 7.9b), defined by

$$A = \mathrm{FR}^{\mathrm{SC}}(t_i, t_{i+1}) \setminus \mathrm{FR}^{\mathrm{TT}}(t_i, t_{i+1})\,. \tag{7.2}$$

Next, it is shown by contradiction that there must exist a successor v_{m+1} of v_m in S_{SC}.

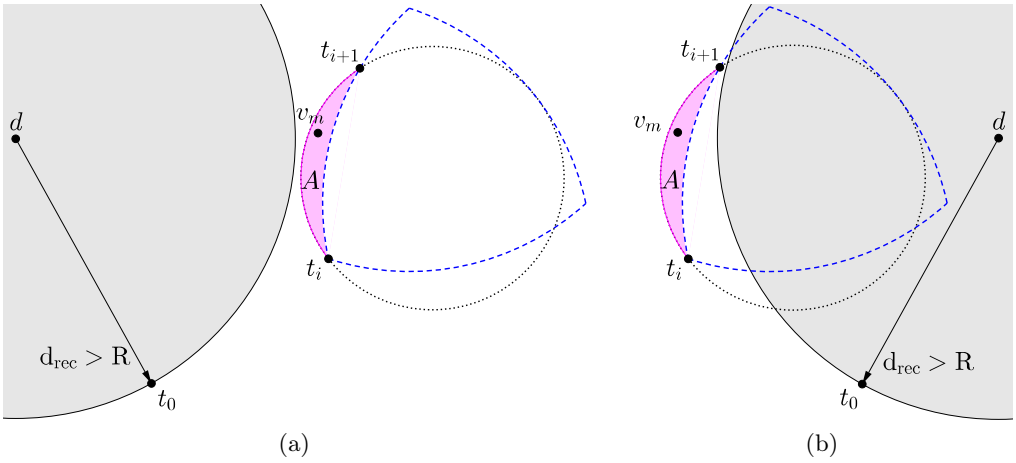


Figure 7.10: (a) represents Case (1), where v_m and d are in the same half-plane w.r.t. $\ell(t_i, t_{i+1})$. (b) represents Case (2), where v_m and d are in different half-planes w.r.t. $\ell(t_i, t_{i+1})$.

For the sake of contradiction, assume v_m is the last node in S_{SC}, i.e., $v_m = v_l$. Then, $\|v_m d\| < \mathrm{d_{rec}}$ must hold, since by Lemma 4 in [68] it holds that RS always terminates at a node with distance to the destination smaller than recovery distance $\mathrm{d_{rec}}$.

Distinguish the cases that v_{j+1} and destination d are located in (1) the same half-plane (see Figure 7.10a), and (2) in different half planes w.r.t. $\ell(t_i, t_{i+1})$ (see Figure 7.10b).

Case (1): If $\mathrm{FR}^{\mathrm{SC}}(v_{j+1}, t_{i+1}) \cap \mathrm{C}_{\mathrm{d_{rec}}}(d) \neq \emptyset$, then according to line 14 of algorithm RS-Shortcut, v_{i+1} would have been selected for forwarding instead of t_{i+1}, which contradicts to the assumption that $v_{j+1} \neq t_{i+1}$ and hence, this case cannot occur.

Now consider the case where $\mathrm{FR}^{\mathrm{SC}}(v_{j+1}, t_{i+1}) \cap \mathrm{C}_{\mathrm{d_{rec}}}(d) = \emptyset$, as given in Figure 7.10a.

If $v_m = v_{j+1}$, then $\|v_m d\| \geq \mathrm{d_{rec}}$ holds because otherwise, during Step 1 of RS-Shortcut, v_{j+1} would have been chosen for forwarding, which contradicts again to the assumption that $v_{j+1} \neq t_{i+1}$.

Otherwise, If $v_m \neq v_{j+1}$, the assumption that $\mathrm{FR}^{\mathrm{SC}}(v_{j+1}, t_{i+1}) \cap \mathrm{C}_{\mathrm{d_{rec}}}(d) = \emptyset$ combined with Equation 7.1 implies that $\|v_m d\| \geq \mathrm{d_{rec}}$ and hence, there must exist a successor v_{m+1} of v_m in S_{SC}.

Case (2): Note that $\|t_i d\| \geq \mathrm{d_{rec}}$, because RS-Shortcut was executed by t_i. Moreover, $\|t_{i+1} d\| \geq \mathrm{d_{rec}}$ must also hold, as otherwise in line 11 of RS-Shortcut node v_{j+1} would have been selected for forwarding, but $t_{i+1} \neq v_{j+1}$. Combined with the facts that v_m and d are on opposite sides of $\ell(t_i, t_{i+1})$, and $v_m \in A$, it holds that $\|v_m d\| \geq \mathrm{d_{rec}}$ and hence there must exist a successor v_{m+1} of v_m in S_{SC}.

Finally, it is shown by contradiction that $v_{m+1} = t_{i+1}$ holds, which concludes this proof, since $v_{m+1} \in S_{\mathrm{SC}}$.

Assume $v_{m+1} \neq t_{i+1}$. By Lemma 7.6, the union of the sweep circle at v_0 in initial position and the sweep areas of the nodes $v_0, ..., v_{m-1}$ is empty of nodes from $V \setminus \{v_0, ..., v_m\}$. Hence,

v_{m+1} must be located outside this empty sweep region, but strictly inside $\mathrm{SC}(w_m, t_{i+1})$ (red area in Figure 7.8).

There are only two possible areas where w_{m+1} may be located, as $\mathrm{TT}(t_i, t_{i+1})$ is empty of nodes.

If w_{m+1} is contained by $\mathrm{SC}(v_m, t_{i+1}) \cap \mathrm{C}_R(t_i)$, then it must be located inside A. This leads to a contradiction, because Σ is assumed to be the longest successive subsequence of nodes from S_{SC}, but in the case under consideration, $\Sigma \oplus w_{m+1}$ would be legitimate and longer subsequence of S_{SC}.

In the remaining case, v_{m+1} must be contained in $\mathrm{SC}(v_m, t_{i+1}) \setminus \mathrm{C}_R(t_i)$. For any point $p \in \mathbb{R}^2$ with $p \in A$ and in particular for v_m, it holds that

$$\mathrm{SC}(p, t_{i+1}) \setminus \mathrm{C_R}(t_i) \subseteq \underbrace{\mathrm{D}(t_{i+1}, t') \setminus \mathrm{C_R}(t_i)}_{\text{critical region incident to } t_{i+1}} ,$$

where t' is the corner of $\mathrm{TT}(t_i, t_{i+1})$ that is opposite to the arc connecting t_i and t_{i+1}. Therefore, v_{m+1} must be located in the critical region incident to t_{i+1}. Then there would have been at least one node $x \in \mathrm{N}_1(t_{i+1})$ responding with a $\mathrm{CTS}(\mathrm{CR}, x, t_{i+1})$ to the corresponding $\mathrm{RTS}(\mathrm{CR}, t_{i+1}, v_{j+1}, t')$ sent by t_{i+1}. In turn, t_{i+1} would have sent a $\mathrm{CTS}(\mathrm{SCut}, t_{i+1}, t_i, 0)$ and t_i would have selected v_{j+1} for forwarding, which contradicts to $t_{i+1} \neq v_{j+1}$. Hence, $t_{i+1} = v_{m+1} \in S_{\mathrm{SC}}$ must hold and thus, $t_{i+1} \in S_{\mathrm{SC}}$. □

7.5 Discussion and Implications

The results presented in the previous sections have several implications.

The first implication concerns so-called *α-shapes* [217], an important graph-theoretical foundation of (local) boundary detection.[5]

Corollary 7.13. *PDT Face traversals are boundaries of (negative) α-shapes.*

Proof. By Lemma 1 in [15], SC traversals are boundaries of (negative) α-shapes. The claim follows from the equivalence of SC and PDT Face traversals (Theorem 7.8). □

Secondly, algorithm RS-Shortcut is of use in geographic unicast routing.

Corollary 7.14. *The combination of Greedy forwarding and algorithm RS-Shortcut guarantees delivery in connected unit disk graphs.*

Proof. This corollary follows directly from the two facts that an RS-Shortcut traversal is a subpath of its corresponding SC traversal (Theorem 7.12), and that the combination of Greedy forwarding and SC traversals guarantees delivery in connected unit disk graphs by Theorem 1 in [15]. □

The next implication concerns the superiority of RS-Shortcut recovery paths over the corresponding SC recovery paths.

[5] For details, the reader may refer to the original work on α-shapes by Edelsbrunner et al. [217] and the discussion on the relation of RS traversals and α-shapes in [15]

Corollary 7.15. *Recovery paths generated by algorithm RS-Shortcut are at most as long as the corresponding* SC *recovery paths, regarding both hop and Euclidean path length. Furthermore, they can help to avoid as many as* $\Theta(n)$ *unnecessary forwarding operations.*

Proof. The first part of the claim follows directly from Theorem 7.12. The second part holds with the example given in Figure 7.5c, illustrating that a single TT hop can avoid as many as $\Theta(n)$ SC recovery hops. □

It is easy to construct scenarios where RS-Shortcut does not perform any better than RS-SC. To see this, consider the graph given in Figure 7.5c: If there exists a node in the critical region incident to v_k, then RS-Shortcut and RS-SC produce equivalent recovery paths. This example highlights in addition that both recovery paths produce as many as $\Theta(n)$ unnecessary forwarding operations, compared to RS-TT. On the other hand, in contrast to TT recovery paths, for recovery paths produced by RS-SC and RS-Shortcut it is known that they belong to a constant stretch Euclidean spanner. In summary, in regard to the set of beaconless recovery algorithms that require only constant number of messages per forwarding step, the paths produced by algorithm RS-Shortcut provide the strongest theoretical guarantees.

RS-SC, RS-TT, and RS-Shortcut have one problem in common: For each of these algorithms, bad-case scenarios such as those depicted in Figure 7.5 can be designed. In these situations the data packet is forwarded multiple times within a single node's transmission radius, i.e., the algorithms perform poorly. The author presented in [77] (in a joint work with Botterbusch and Frey) a simple but promising idea to determine and avoid such situations. Instead of immediate data forwarding after detection of the next-hop, a lightweight probing packet is used to explore the recovery path within the current forwarding node's transmission range. This way, a locally optimal hop (w.r.t. some user-defined optimization function) for actual data transmission is determined and used. For example, the best hop for forwarding on the recovery path according to some energy optimization criterion can be selected. Simulations suggest that, independent of the node density, this extension can lead to significant reductions of data packet transmissions, and reduced energy consumption, compared to immediate data transmission.

Chapter 8

Reactive Local Construction of Topological Quasi Unit Disk Graph Spanners

When replacing the simplistic unit disk by the more realistic quasi unit disk graph model, the reactive local topology control and recovery algorithms rPDT and RS-Shortcut, introduced in the previous chapters, are no longer guaranteed to operate correctly. This results from the QUDG properties (summarized as Facts/Lemmata 3.10– 3.15 in Section 3.2), in particular from the fact that local planarization techniques like Gabriel graph and partial Delaunay triangulation possibly yield disconnected outputs if applied onto QUDGs. In fact, it is in general impossible to extract planar and connected subgraphs of QUDGs, while the maximum and minimum transmission radii of the nodes differ, i.e., while $r < R$. The survey on QUDG planarization techniques in Section 3.2 reveals two common solutions to this problem, given that $R/r \leq \sqrt{2}$. Either the QUDG is enriched by adding virtual edges, which guarantee connectivity when applying standard UDG planarization techniques, or a sparse connected backbone graph is extracted, which is then planarized, e.g., by replacing edge intersections by virtual nodes.

All of the existing approaches have in common that they are beacon-based in the sense that they either make active use of beaconing for construction of nodes' neighborhood tables, or assume nodes to be provided with a priori k-neighborhood knowledge, where $k \geq 1$. To the best of the author's knowledge, presently there are neither beaconless algorithms for planarization of QUDGs, nor beaconless guaranteed delivery geographic routing algorithms for such graphs.

In this chapter, this research gap is closed. The main contribution is an Ω-reactive local topology control algorithm, called *ReactiveBackbone*. It allows a node from a QUDG, satisfying $R/r \leq \sqrt{2}$, to compute its local view on a constant stretch topological spanner of the input graph locally and without use of beaconing. Based on this algorithm, a guaranteed delivery geographic routing protocol called *Reactive-Virtual-Face-Traversal* is introduced. These algorithms are based on beacon-based planarization and geographic routing techniques introduced by Lillis et al. [70, 71] and Guan [72, 73], respectively.

Lillis et al. present the only local solution for distributed construction of planar QUDG spanners. During execution of their algorithm the nodes maintain neighborhood tables and exchange messages with their 3-hop neighbors. Given a QUDG G satisfying $R/r \leq \sqrt{2}$, their algorithm computes a virtual planar backbone graph $\mathrm{Virt}(G_b)$ for G by planarizing

a sparse backbone graph G_b via introduction of virtual nodes. The connection of any non-backbone node with its cluster head in $\mathrm{Virt}(G_b)$ yields the routing graph G_r. The latter is a constant stretch hop spanner for G and facilitates local geographic routing with guaranteed delivery (see Section 8.1.2 for further details). However, this algorithm is designed as a synchronous, distributed algorithm for construction of the topology over the entire network graph and assumes synchronous wake-up of nodes. That is, all nodes start execution of the algorithm proactively and synchronously.

The local geographic routing protocol *Virtual-Face-Traversal* by Guan [72, 73] (see Section 3.2 for details) also uses the virtual nodes idea to guarantee delivery on connected QUDGs satisfying $\mathrm{R/r} \leq \sqrt{2}$. It does not require topology control operations. However, for computation of the next forwarding decision, a node has to be provided with complete 2-hop neighborhood information. Since this protocol operates directly on the network graph rather than on some controlled topology, its performance regarding routing stretch and message overhead may be poor, at least in worst-case scenarios.

As a first major contribution, in Section 8.2 it is shown that the topology mapping defined by the algorithm from Lillis et al. is in fact a k-local topology mapping for $k = 8$. This implies that for the construction of a node's local view on the virtual planar backbone graph $\mathrm{Virt}(G_b)$ it suffices to execute the distributed algorithm by all nodes in the 8-hop neighborhood of that node. Then it is shown that it is in fact even sufficient to construct the backbone only by those nodes that belong to cells of a geographical clustering that are at most six hops apart and belong to a specific geographic region, called 3-cell neighborhood. Based on these results, in Sections 8.3 & 8.4, the Ω-reactive local topology control algorithm *ReactiveBackbone* is derived, which consists of multiple parts that are designed and explained successively. This algorithm is analyzed regarding its message complexity and shown to be Ω-reactive in Section 8.5. In Section 8.6 the guaranteed delivery reactive geographic routing protocol *Reactive-Virtual-Face-Traversal* is introduced. This chapter closes with a discussion and further implications in Section 8.7.

8.1 Preliminaries

8.1.1 Model, Assumptions, and Notations

In the following, network graphs are modeled as quasi unit disk graphs $G = (V, E)$ over finite and distinct node sets $V \subset \mathbb{R}^2$, where $|V| = n$. It is assumed that nodes are provided with their geographic positions, are aware of the maximum and minimum transmission radii R and r, respectively, and that $\mathrm{R/r} \leq \sqrt{2}$. The latter implies that the maximum hop distance between the endpoints of two intersecting edges $uv, xy \in E$ is at most three (see Fact 3.13). For convenience, QUDGs satisfying $\mathrm{R/r} \leq \sqrt{2}$ are henceforth also referred to as *restricted QUDGs*.

As in previous chapters, the beaconless algorithms introduced here use a delay function for delaying the nodes' operations (e.g., sending of a message) by a fraction or multiple of time slice $\mathrm{t_{max}} > 0$, an arbitrary but fixed constant known to all nodes. The delay

of a node v is proportional to some Euclidean distance δ that typically depends on v's geographic position. This is modeled by the following delay function:

$$t(\delta, \Delta) = (\delta/\Delta) \cdot \mathrm{t}_{\max}\,, 0 < \delta \leq \Delta\,.$$

As usual in beaconless algorithms it is assumed that messages are transmitted instantly and reliably. The problems arising with this simplifying assumption are dicussed in a more general context in Section 10.2 as part of the conclusion.

8.1.2 Distributed Construction of the Virtual Planar Backbone Graph

The following is a detailed description of the distributed algorithm for construction of $\mathrm{Virt}(G_b)$ and routing graph G_r by Lillis et al. [70, 71].

The plane is partitioned by virtually placing an infinite, axes-parallel square grid with one grid vertex in the origin $(0, 0)$ and grid cell diameter r (see Figure 8.1a). Under the assumption that nodes know their geographic positions in a global coordinate system as well as radius r, each node can compute to which *grid cell* (or simply *cell*) it belongs. The grid cell in which a node u is located is addressed by $C(u)$. This gives a unique *geographical clustering* of the nodes in the underlying QUDG $G = (V, E)$.[1] Since the diameter of the grid cells is r, all nodes within a particular cell form a clique in G and the transmission of a node is always overheard by all other nodes in that cell.

At first, all nodes in V locally broadcast their positions such that all nodes know their one-hop neighborhoods. Given this information, without further communication all nodes in a particular cell C can deterministically agree on a cluster head, e.g., by choosing the node closest to the geometric center of C, or the one having the smallest node ID (if available). Then all nodes in V locally broadcast the position of their cluster heads. Given this information, each node computes the set of distinct cluster heads of its neighboring nodes and locally broadcasts this set. Now the cluster head of each cell C knows to which other cells C_j it is connected by at least one edge in E. For each such neighboring cell, the cluster head chooses exactly one *representative bridge edge*[2]. The cluster heads locally broadcast their sets of representative bridge edges. Endpoints of representative bridge edges inform the corresponding other endpoint about this choice via a local broadcast.[3] Now every node in V knows whether it is a cluster head, bridge edge endpoint, or neither

[1] This is a standard technique used also in [86, 150]

[2] Lillis et al. do not further specify how a representative bridge edge is chosen. In fact the choice is irrelevant for proving the spanning ratio. The latter also holds for the choice of a cell's cluster head. However, in order to be able to compare the output graphs produced by the distributed algorithm by Lillis et al. and the beaconless approach presented here, the following selection rules are defined: Cluster head of a cell C is a node in C that is closest to the geometric center of C regarding Euclidean distance. For two cells C, Z, let E_{CZ} be the set of edges connecting nodes in C to nodes in Z; then the cluster head of C selects as representative bridge edge for Z an edge $cz \in E_C$, where $c \in C$ and $z \in Z$, such that both c and z minimize the Euclidean distance to the geometric center of Z among all edges in E_{CZ}.

[3] Lillis et al. do not mention this last step in [70, 71]. However, it is necessary in order to ensure correctness of the approach, since otherwise some bridge edge endpoints are never informed about their assigned role.

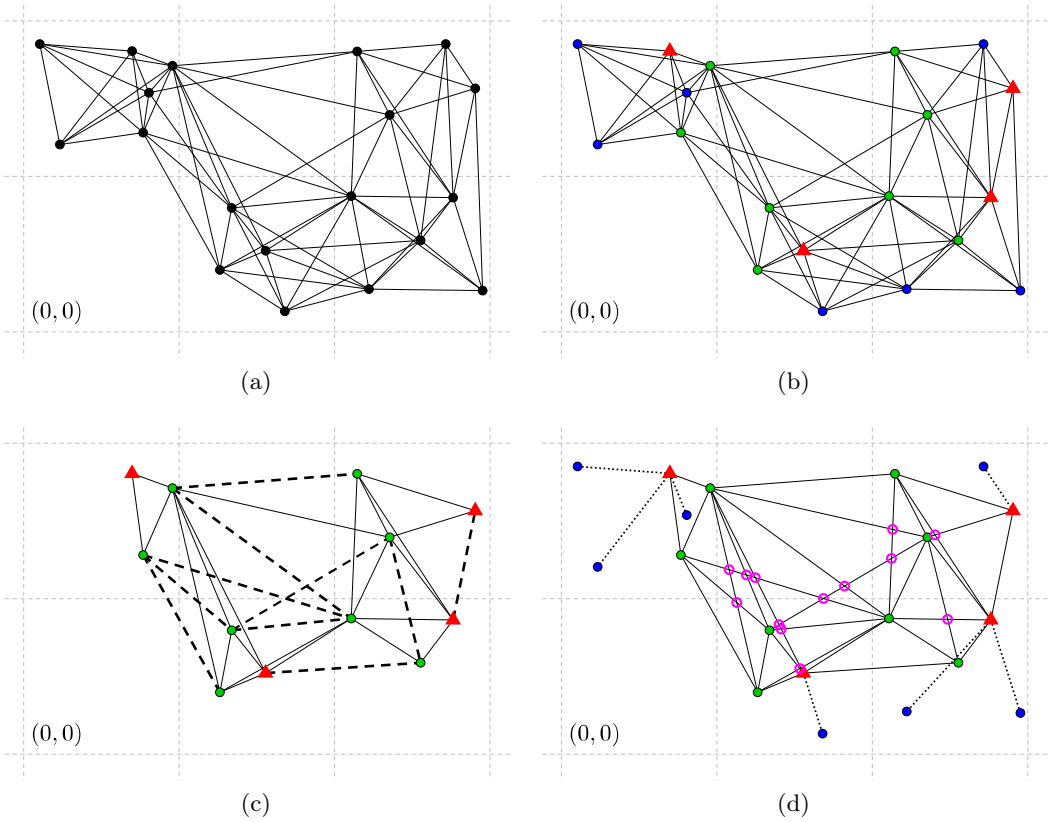


Figure 8.1: (a) Input QUDG $G = (V, E)$. (b) Cluster heads are red triangles and endpoints of bridge edges are green dots. (c) Backbone graph $G_b = (V_b, E_b)$ with induced edges given by the dashed lines. (d) Routing graph G_r, where dotted lines connect non-backbone nodes (blue) with their cluster heads; pink circles represent virtual nodes at edge intersections of edges in G_b. This figure is similar to Figure 8 in [71].

of it (see Figure 8.1b). The set of cluster heads and bridge edge endpoints is then the set of backbone nodes $V_b \subseteq V$. One final local broadcast by all nodes in V_b suffices so that each node knows its adjacency in the induced graph $G_b = (V_b, E_b)$ w.r.t. G, i.e., the graph with node set V_b and edge set $E_b = \{uv \in E : u, v \in V_b\}$ (see Figure 8.1c).

Backbone graph G_b has the following properties.

(i) The node degree of any node in G_b is $\mathcal{O}(1)$.

(ii) Each edge in E_b is intersected at most $\mathcal{O}(1)$ times.

(iii) V_b is a connected dominating set for G.

Property (i) holds for the following reason. By Lemma 4 in [150], it holds that a cell C is connected to at most $\mathcal{O}(\mathrm{R}^2/\mathrm{r}^2)$ other cells. Under the assumption that $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$, this is

a constant upper bound. This implies in particular that the 3-hop neighborhood of any backbone node in V_b is of size $\mathcal{O}(1)$. For proofs of the other two properties, see Lemma 4 in [71].

Now the backbone graph G_b is transformed into a planar graph $\mathrm{Virt}(G_b) = (V_{\mathrm{Virt}}, E_{\mathrm{Virt}})$ using an idea previously described in [84, 86]. V_{Virt} contains all nodes from V_b and in addition, for each edge intersection of two edges $uv, xy \in E_b$, it contains a *virtual node* w whose position is that of the corresponding edge intersection (pink circles in Figure 8.1d). Edge set E_{Virt} is composed of all edges in E_b with the following modifications: Any two intersecting edges uv, xy are deleted and edges uw, vw, xw, yw are added.

The distributed construction of the backbone nodes' adjacencies in $\mathrm{Virt}(G_b)$ is straightforward. All backbone nodes gather 3-hop neighborhood information in G_b and then compute their adjacency in E_{Virt} without further communication. This is feasible for the following reason. Since V_b is a connected dominating set for G by property (iii), G_b is a connected QUDG if G is a connected QUDG. Since G satisfies $\mathrm{R/r} \leq \sqrt{2}$ this also holds for G_b. While a node in V_b is provided with 3-hop neighborhood knowledge of G_b, it knows all endpoints of edges in E_b that are intersecting its incident edges. Therefore, without further communication it can compute the positions of the corresponding virtual nodes as well as its adjacency in E_{Virt}.

Finally, define the *routing graph* G_r as $\mathrm{Virt}(G_b)$ combined with all non-backbone nodes $V \setminus V_b$ connected to their corresponding cluster heads by an edge from E (see Figure 8.1d). The following theorem follows immediately from the proofs of Lemmata 4 & 7 in [71].

Theorem 8.1 (Topological spanning ratio of G_r)**.** *Let $G = (V, E)$ be any connected QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$. G_r is a hop $(4c + 6)$-spanner of G, where $c \in \mathcal{O}(1)$ denotes the maximum number of intersections of any edge in G_b.*

Lillis et al. observe that the routing graph G_r is not a constant stretch Euclidean spanner for the underlying QUDG $G = (V, E)$. This holds for the following reason. Let $u, v \in V$ be two non-backbone nodes which are connected in G by an edge, but which belong to different, neighboring grid cells. While moving these nodes closer to the grid cell boundary separating their cells, their Euclidean distance can be made arbitrarily small and hence, the Euclidean spanning ratio w.r.t. G_r becomes arbitrarily large. Therefore, Lillis et al. propose the following solution. Instead of only one grid, three grids (blue, red, and green) of grid cell diameter at most r are placed on the plane. The blue grid passes through the origin, whereas the other two are shifted by 1/3 and 2/3 of the grid cell diameter to the right. In each grid cell of any color, the Euclidean spanner BPS_2 [131, 132] (see Section 3.1) is constructed over the clique of nodes induced by this cell. The routing graph G'_r then consists of $\mathrm{Virt}(G_b)$ constructed over the blue grid, as well as of all Euclidean spanners. Routing graph G'_r is then proven to be a constant stretch topological and constant stretch Euclidean spanner for the underlying QUDG.

8.2 On the Local Constructibility of the Virtual Planar Backbone Graph

In the following, two major contributions are derived. Firstly, in Section 8.2.1 it is proven that the topology mapping described by Lillis et al. [70, 71] on how to transform a given QUDG G into a virtual planar backbone graph $\mathrm{Virt}(G_b)$ is in fact a k-local topology mapping w.r.t. Definition 5.2. Subsequently, in Section 8.2.2 it is shown that for the construction of the local view of any real or virtual backbone node in $\mathrm{Virt}(G_b)$ it is sufficient to construct the backbone in all cells which are at most a constant number of hops apart and belong to a bounded geographic region around this node. This result enables the design of the Ω-reactive local topology control algorithm ReactiveBackbone in subsequent sections.

8.2.1 Local Topology Mapping

For simplicity and comprehensibility, the following notation is introduced. Given a QUDG $G = (V, E)$, $\tau(G)$ refers to the virtual planar backbone graph $\mathrm{Virt}(G_b)$ constructed over the entire network graph G. Given any node $u \in V$, $G[u, k]$ refers to the k-hop local view of u on graph G, which is the set of nodes from V reachable from u via at most k hops, i.e., $\mathrm{N}_k(u)$, and all edges $\{xy \in E : x \in \mathrm{N}_{k-1}(u) \text{ and } y \in \mathrm{N}_k(u)\}$. Then $\tau(G[u, k])$ refers to the virtual planar backbone construction over graph $G[u, k]$. For a real node $u \in V$ it is said to be a *generator* of virtual node w, if u is a backbone node and one of its incident backbone edges intersects at least one other backbone edge in w.

The main theorem requires the following auxiliary lemmata.

Lemma 8.2 (Lemma 2 in [71])**.** *Let $u, v \in \tau(G)$ be any two neighboring virtual nodes. The generators of u and v in $\tau(G)$ are at most 5 hops apart from each other in G.*

For an illustration, see Figure 8.3b.

Lemma 8.3. *A node $v \in V$ belongs to $\tau(G)$, if and only if v belongs to $\tau(G[v, 3])$.*

Proof. It trivially holds that v is cluster head of its cell in $\tau(G)$, if and only if it is cluster head in $\tau(G[v, 3])$ since all nodes in cell $C(v)$ are 1-hop neighbors of v and therefore belong to $G[v, 3]$. If v is not a cluster head, then it is a bridge node, either selected by the cluster head of its cell $C(v)$, $\mathcal{H}(C(v))$, or a cluster head of an adjacent cell Z. In the former case, the decision of selecting v as a bridge node by $\mathcal{H}(C(v))$ in $\tau(G)$ only depends on the nodes in $C(v)$ and their incident edges, all of which are contained by $G[v, 2]$ and hence, an equivalent decision is made in $\tau(G[v, 3])$. In the latter case, the decision by the cluster head $\mathcal{H}(Z)$ is based on all nodes in Z as well as their incident edges. The nodes in Z are 2-hop neighbors of v and their incident edges connect to nodes that are at most 3 hops apart from v. Hence, these nodes and edges are contained by $G[v, 3]$ and thus, an equivalent decision is made in $\tau(G[v, 3])$. □

Theorem 8.4. *The topology control mapping described by Lillis et al. [70, 71] is a k-local topology mapping for $k = 8$.*

Proof. It needs to be shown that for any restricted QUDG graph $G = (V, E)$ and for all $v \in \tau(G)$, there exits a node $u \in V$, such that $\tau(G)[v] = \tau(G[u, 8])[v]$.

In the following, let $G = (V, E)$ be any restricted QUDG, and $v \in \tau(G)$ any node.

"$\Rightarrow$": It has to be shown that there exists $u \in V$ such that for an arbitrary but fixed neighbor $w \in \tau(G)[v]$ it holds that $w \in \tau(G[u, 8])[v]$.

Case 1 (both v and w are real nodes): It holds that $vw \in E$ and hence, $w \in G[v, 1]$ and $G[w, 3] \subset G[v, 8]$. Then, by Lemma 8.3 it holds that w is a backbone node in $\tau(G[v, 8])$. Assume $w \notin \tau(G[v, 8])[v]$, then there must exist a backbone edge xy in $\tau(G[v, 8])$ that intersects vw. This implies that x and y are at most 3 hops apart from v. Hence, $G[x, 3], G[y, 3] \subset G[v, 8]$ and by Lemma 8.3 both x and y are also backbone nodes in $\tau(G)$, and the edge xy intersects vw in $\tau(G)$. But this contradicts the assumption that vw is not intersected in $\tau(G)$, hence, $w \in \tau(G[v, 8])[v]$, and with $u = v$ an appropriate node exists.

Case 2 (v is a real node, w is a virtual node): In $\tau(G)$, virtual node w is generated by some backbone edge vu, and one or several other backbone edges. Let xy be any such backbone edge. It holds that u, x, y are at most 3 hops apart from v in G and hence, $G[u, 3], G[x, 3], G[y, 3] \subset G[v, 8]$. Then, by Lemma 8.3 it holds that u, x, y are backbone nodes in $\tau(G[v, 8])$ and hence, $w \in \tau(G[v, 8])$. Using the same argument as in Case 1, it cannot be the case that vw is intersected by another backbone edge in $\tau(G[v, 8])$. Thus, $w \in \tau(G[v, 8])[v]$ holds and with $u = v$ an appropriate node exists.

Case 3 (v is a virtual node, w is a real node): This case is symmetric to Case 2 and with $u = w$, there exists a node such that $w \in \tau(G[u, 8])[v]$.

Case 4 (both v and w are virtual nodes): Because v and w are neighbors in $\tau(G)$, it holds that they are generated by one common backbone edge xy. Let ab and cd be arbitrary backbone edges that intersects xy in v and w, respectively. By Lemma 8.2, the hop distance between any two generators $p, q \in \{x, y, a, b, c, d\}$, $p \neq q$, is at most 5. Hence, it holds that $G[q, 3] \subset G[p, 8]$. Now Lemma 8.3 implies that all nodes in $\{x, y, a, b, c, d\}$ are also backbone nodes in $\tau(G[p, 8])$ which implies in particular that the virtual nodes v and w exist in $\tau(G[p, 8])$. Using the same argument as in Case 1, it cannot be the case that vw is intersected by another backbone edge in $\tau(G[p, 8])$ and hence, $w \in \tau(G[p, 8])[v]$ holds and with $u = p$ an appropriate node exists.

"$\Leftarrow$": It has to be shown that there exists $u \in V$, such that for an arbitrary but fixed neighbor $w \in \tau(G[u, 8])[v]$ it holds that $w \in \tau(G)[v]$.

Case 1 (v is a real node): By assumption $v \in \tau(G)$ holds and hence, by Lemma 8.3 v is also a backbone node in $\tau(G[v, 8])[v]$. Let $w \in \tau(G[v, 8])[v]$ be any neighbor of it.

Subcase 1.1 (w is a real node): With $vw \in E$, $G[w, 3] \subset G[v, 8]$ holds and Lemma 8.3 yields that w is a backbone node in $\tau(G)$. If $w \notin \tau(G)[v]$, then there exists a backbone edge xy in $\tau(G)$ which intersects vw. The latter implies that x, y are at most 3 hops apart from v in G. Hence, $G[x, 3], G[y, 3] \subset G[v, 8]$ and by Lemma 8.3 it holds that both x and y are also backbone nodes in $\tau(G[v, 8])$. But then, xy and vw would also intersect in $\tau(G[v, 8])$, which contradicts the assumption that vw are neighbors in that graph.

Subcase 1.2 (w is a virtual node): Let w be generated by the backbone edges uv and xy in $\tau(G[v, 8])$. Since u, x, y are at most 3 hops apart from v in G, $G[u, 3], G[x, 3], G[y, 3] \subset G[v, 8]$ holds. By Lemma 8.3, u, x, and y are also backbone nodes in $\tau(G)$ and thus, w is

a virtual node in $\tau(G)$. Using the same argument as in Subcase 1.1, it cannot be the case that vw is intersected by another backbone edge in $\tau(G)$. Hence, $w \in \tau(G)[v]$ holds.

In both subcases an appropriate node u, namely v, is shown to exist.

Case 2 (v is a virtual node): Consider $\tau(G[u, 8])$, where u is any generator of v in $\tau(G)$. Lemma 8.3 implies that u, as well as all other generators of v in $\tau(G)$ (which are at most 3 hops apart from u in G), are also backbone nodes in $\tau(G[u, 8])$. Hence, v is a virtual node in $\tau(G[u, 8])$. Let $w \in \tau(G[u, 8])[v]$ be any node.

Subcase 2.1 (w is a real node): Backbone node w must be a generator of v, for which it has already been shown that they are backbone nodes in $\tau(G)$. If $w \notin \tau(G)[v]$, then there exists a backbone edge xy in $\tau(G)$ which intersects vw. By Lemma 8.2, x and y are at most 5 hops apart from u in G and hence, $G[x, 3], G[y, 3] \subset G[u, 8]$. Lemma 8.3 implies that x and y are also backbone nodes in $\tau(G[u, 8])$. But then, xy and vw would also intersect in $\tau(G[u, 8])$, which contradicts the assumption that vw are neighbors in that graph. Hence, $w \in \tau(G)[v]$ must hold.

Subcase 2.2 (w is a virtual node): Virtual nodes v and w are generated in $\tau(G[u, 8])$ by a common backbone edge xy and at least two other backbone edges ab and cd. Moreover, it holds that $u \in \{x, y, a, b, c, d\}$. By Lemma 8.2 it holds that any of these generators is at most 5 hops apart from u in G and hence, for any $p \in \{x, y, a, b, c, d\}$, it holds that $G[p, 3] \subset G[u, 8]$. Then, Lemma 8.3 implies that p is also a backbone node in $\tau(G)$ and therefore, both virtual nodes v and w exist in $\tau(G)$. Using the same argument as in Subcase 2.1, it cannot be the case that vw is intersected by another backbone edge in $\tau(G)$, and hence $w \in \tau(G)[v]$ holds.

In both subcases a suitable node u is shown to exist, which concludes the proof. □

8.2.2 Local View Construction

The above theorem states that for any $v \in \text{Virt}(G_b)$ there exists a node $u \in V$, such that the local view of v on $\text{Virt}(G_b)$ can be constructed if all nodes in $G[u, 8]$ execute the algorithm by Lillis et al.

In the following it is shown that it is in fact not necessary that all 8-hop neighbors of u participate in the local view construction, but only those which belong to a specific geographic region around u, called 3-cell neighborhood of u (defined shortly). More specifically, it is shown that it is sufficient that backbone nodes are constructed by all nodes $w \in V$ for which the following holds: (1) Cell $C(w)$ is contained in the 3-*cell neighborhood* of u. (2) There exists at least one node in $C(w)$ which is connected to u by a path consisting of at most 6 hops.[4] (3) This path visits only nodes that are contained in the 3-cell neighborhood of u. The latter is defined as follows.

Definition 8.5 (k-cell neighborhood)**.** Two grid cells C and Z are said to be *neighboring cells*, if there exist a line segment $\overline{pq}$ of length at most R, s.t. $p \in C$ and $q \in Z$. The

[4] Note that this condition still reflects the constant $k = 8$ proven for the topology control mapping described by Lillis et al. A node w whose cell is connected to u by a path of hop-length 6 is itself at most 7 hops apart from u. Bridge edge selection requires each node to consider its 1-hop neighborhood and hence, this requires information about 8-hop neighbors of u.

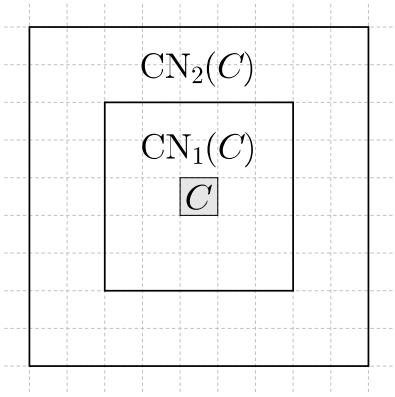


Figure 8.2: Illustration of Definition 8.5 of cell neighborhoods of a particular cell C, where $\mathrm{CN}_1(C)$ and $\mathrm{CN}_2(C)$ are given by the set of cells enclosed by the inner and outer square, respectively.

1-*cell neighborhood* of a cell C, denoted $\mathrm{CN}_1(C)$, is the set of all neighboring cells of C, including C itself. The k-*cell neighborhood* of C for $k > 1$ is then given by

$$\mathrm{CN}_k(C) = \bigcup_{Z \in \mathrm{CN}_{k-1}(C)} \mathrm{CN}_1(Z)\,.$$

See Figure 8.2 for an illustration.

Lemma 8.6. *For a QUDG $G = (V, E)$ satisfying $\mathrm{R/r} \leq \sqrt{2}$ and two intersecting edges $uv, xy \in E$ it holds that*

$$\{C(v), C(x), C(y)\} \subseteq \mathrm{CN}_2(C(u))\,.$$

Proof. Since $\|uv\| \leq \mathrm{R}$, $C(v) \in \mathrm{CN}_1(C(u))$. Now, consider the quadrangle formed by u, v, x, y. At least one of the internal angles is at least $\pi/2$. Assume w.l.o.g. that $\angle xvy \geq \pi/2$. Then $v \in \mathrm{D}(x, y)$. Since $\|xy\| \leq \mathrm{R}$, it holds that $\|xv\|, \|yv\| \leq \mathrm{R}$. Hence, $C(x)$ and $C(y)$ belong to $\mathrm{CN}_1(C(v))$ which is a subset of $\mathrm{CN}_2(C(u))$. □

Definition 8.7 (Reachable cell)**.** Let C be any non-empty grid cell and $Z \in \mathrm{CN}_k(C)$. Cell Z is said to be *reachable* w.r.t. $\mathrm{CN}_k(C)$, if there exists a path $\Pi_G(c, z)$ between any node $c \in C$ and $z \in Z$ in G, s.t. all nodes in $\Pi_G(c, z)$ are contained by $\mathrm{CN}_k(C)$.

Lemma 8.8. *Let G be a QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$ and $Z \in \mathrm{CN}_2(C)$ be a cell which is not reachable w.r.t. $\mathrm{CN}_2(C)$ in G. No edge in G with at least one endpoint in Z can intersect with an edge from G that has at least one endpoint in C.*

Proof. For the sake of contradiction assume that Z is not reachable w.r.t. $\mathrm{CN}_2(C)$, but there are two intersecting edges $uv, xy \in E$, with $x \in Z$ and $u \in C$. It holds that at least one of the nodes in $\{x, y\}$ is connected to one of the nodes in $\{u, v\}$ by an edge in G. Assume w.l.o.g. that $yv \in E$. Lemma 8.6 implies that $\{C(u), C(v), C(x), C(y)\}$ is

a subset of $\mathrm{CN}_2(C)$. But then $\Pi_G(u,x) = \langle u,v,y,x\rangle$ is a path from u to x for which it holds that all nodes belong to $\mathrm{CN}_2(C)$ and thus, Z is reachable w.r.t. $\mathrm{CN}_2(C)$, which is a contradiction. □

The above lemma implies that edges in the backbone graph G_b with at least one endpoint in cell C can only be intersected by other edges in G_b, whose endpoints are contained by cells in $\mathrm{CN}_2(C)$ and that are reachable w.r.t. $\mathrm{CN}_2(C)$. Hence, to construct the local view of a real node u in C in the virtual planar backbone graph, it is in fact sufficient to construct all backbone nodes in all reachable cells in $\mathrm{CN}_2(C)$.

Lemma 8.9. *Given a QUDG $G = (V,E)$ satisfying $\mathrm{R/r} \leq \sqrt{2}$ and the corresponding virtual planar backbone graph $\mathrm{Virt}(G_b)$. For a node $v \in V$ it holds that $v \in \mathrm{Virt}(G_b)$ if and only if cluster head and bridge edge construction has been performed at least in $C(v)$, as well as in any reachable cell w.r.t. $\mathrm{CN}_1(C(v))$.*

Proof. The construction of $\mathrm{Virt}(G_b)$ is essentially nothing other than cluster head and bridge edge selection in each non-empty cell. Moreover, any node $v \in V$ belongs to $\mathrm{Virt}(G_b)$ if it is cluster head of its cell, or if it is a bridge node selected by its cluster head, or the cluster head of a reachable cell in $\mathrm{CN}_1(C(v))$. Hence, cluster head and bridge edge selection by non-reachable cells or cells outside $\mathrm{CN}_1(C(v))$ have no influence on whether or not v is a node in $\mathrm{Virt}(G_b)$. □

The combination of Lemmata 8.8 & 8.9 has the following implication. Given a constrained QUDG $G = (V,E)$, then for the local view construction on $\mathrm{Virt}(G_b)$ for any node $u \in V$ it is sufficient to perform cluster head and bridge edge selection in all reachable cells w.r.t. $\mathrm{CN}_3(C(u))$. This holds since cluster head and bridge edge selection in all reachable cells w.r.t. $\mathrm{CN}_3(C(u))$ is sufficient to construct all backbone nodes from G_b located in all reachable cells w.r.t. $\mathrm{CN}_2(C(u))$, by Lemma 8.9. Given these backbone nodes, for any real backbone node v in $C(u)$, all intersections of an incident backbone edge vw are present, since it can only be intersected by backbone edges whose endpoints are reachable w.r.t. $\mathrm{CN}_2(C(u))$ by Lemma 8.8.

In Theorem 8.12 it is now shown that for any real or virtual node $w \in \mathrm{Virt}(G_b)$ there always exists a node u such that it can compute the local view of w as long as cluster head and bridge edge selection is performed in all cells in $\mathrm{CN}_3(C(u))$ which are connected to u in G by a path of at most 6 hops. This result requires the following two auxiliary lemmata.

Lemma 8.10. *Let $G = (V,E)$ be a QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$ and let $E_p \subseteq E$ be the subset of edges from E that share a common point $p \in \mathbb{R}^2$, i.e., all edges in E_p intersect each other in p. For the endpoint u of edges in E_p that is closest to p regarding Euclidean distance (with ties broken arbitrarily), it holds that $u \in \mathrm{D}(x,y)$, for all edges $xy \in E_p$.*

Proof. Let v denote the other endpoint of the edge in E_p whose endpoint is u. By assumption it holds that no three nodes in V are collinear. Hence, $u \neq p$. Moreover it trivially holds that $u \in \mathrm{D}(u,v)$. For any other edge $xy \in E_p$ it holds that $\|xp\|, \|yp\| \geq \|up\|$. Hence, circle $\mathrm{C}_{\|up\|}(p)$ is completely contained by $\mathrm{D}(x,y)$. Since xy was chosen arbitrarily, this holds for all edges in E_p. □

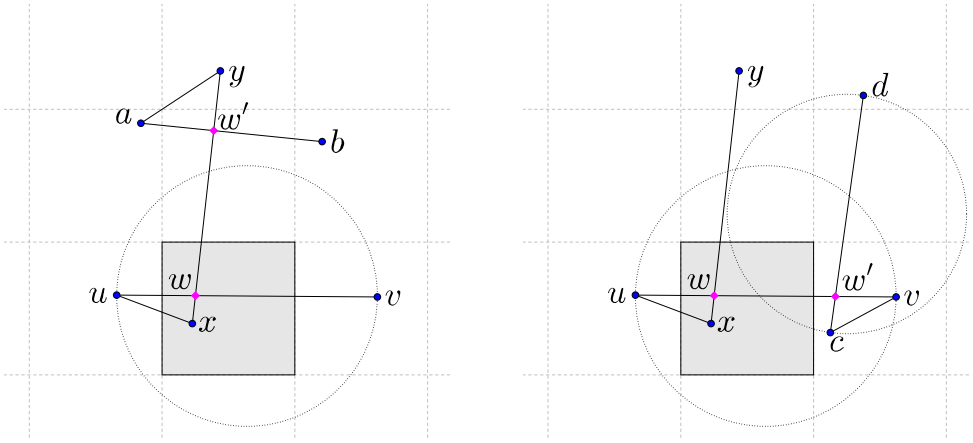


Figure 8.3: Illustrations for the proof of Lemma 8.11. Disks are given by dotted circles. (a) represents Case 1, where w' is a virtual node on xy. (b) represents Cases 2a and 2b, where w' is a virtual node not generated by xy.

Lemma 8.11. *Given a QUDG $G = (V, E)$ satisfying $\mathrm{R/r} \leq \sqrt{2}$, let $w \in \mathrm{Virt}(G_b)$ be any virtual node and $\mathrm{N}_1^{\mathrm{Virt}(G_b)}(w)$ the 1-hop neighborhood of w in $\mathrm{Virt}(G_b)$ (including w itself). There exists a generator $x \in V$ of w, s.t. all real nodes in $\mathrm{N}_{\mathrm{Virt}(G_b)}(w)$, as well as all generators of all virtual nodes in $\mathrm{N}_{\mathrm{Virt}(G_b)}(w)$ are located in reachable cells w.r.t. $\mathrm{CN}_2(C(x))$, and these nodes are at most 4 hops apart from x in G.*

Proof. Let E_w denote the set of backbone edges in $\mathrm{Virt}(G_b)$ also contained in E, and which contain point w. Let $xy \in E_w$ be an edge, such that x is closest to w, regarding Euclidean distance, w.r.t. all endpoints of edges in E_w (ties can be broken arbitrarily). By Lemma 8.10 it holds that $x \in \mathrm{D}(u, v)$, for all $uv \in E_w$.

It trivially holds that $C(x)$ and $C(y)$ are reachable cells w.r.t. $\mathrm{CN}_2(C(x))$ because y is a 1-hop neighbor of x in G.

Next, consider an arbitrary real neighbor $u \in \mathrm{N}_1^{\mathrm{Virt}(G_b)}(w)$. This node is a neighbor of w, since one of its incident backbone edges uv intersects xy in w. By Lemma 8.6 it holds that $C(u)$ and $C(v)$ are contained in $\mathrm{CN}_2(C(x))$. Moreover, it holds that u and v are at most 3 hops apart from x in G. Hence, $C(u)$ and $C(v)$ are reachable w.r.t. $\mathrm{CN}_2(C(x))$.

It remains to show that the claim holds for all other generators of virtual nodes in $\mathrm{N}_1^{\mathrm{Virt}(G_b)}(w)$. Let w' be any such virtual node.

Case 1 (w' is a point on xy): For an illustration, see Figure 8.3a. There exists at least one other backbone edge ab that intersects xy in w'. By Lemma 8.6 it holds that $C(a)$ and $C(b)$ are contained in $\mathrm{CN}_2(C(x))$. Moreover, it holds that both a and b are at most 3 hops apart from x in G. Hence, $C(a)$ and $C(b)$ are reachable w.r.t. $\mathrm{CN}_2(C(x))$.

Case 2 (w' is not a point on xy): For an illustration, see Figure 8.3b. Virtual node w' is generated by a backbone edge uv which intersects xy in w, as well as one or several backbone edges which intersect uv in w'. Consider a fixed but arbitrarily chosen edge cd

intersecting uv in w'. By choice of x it holds that $x \in \mathrm{D}(u,v)$. Since $\|uv\| \leq \mathrm{R}$ it holds that $\|xu\|, \|xv\| \leq \mathrm{R}$. At least one of the internal angles of the quadrangle formed by $\{u,v,c,d\}$ is at least $\pi/2$.

Subcase 2.1 ($\measuredangle ucv \geq \pi/2$ or $\measuredangle udv \geq \pi/2$): Assume w.l.o.g. that $\measuredangle ucv \geq \pi/2$ (the other case is symmetric). Then, both x and c are contained in $\mathrm{D}(u,v)$ and with $\|uv\| \leq \mathrm{R}$ it holds that $\|xc\| \leq \mathrm{R}$. Since cd is an edge from the input graph, it holds in addition that $\|cd\| \leq \mathrm{R}$. Application of the triangle inequality then yields

$$\|xd\| \quad \leq \quad \|xc\| + \|cd\| \leq \mathrm{R} + \mathrm{R} = 2\mathrm{R}\,.$$

Hence, $C(c)$ and $C(d)$ belong to $\mathrm{CN}_2(C(x))$. It also holds that x is at most 2 hops apart from u and v in G (since $x \in \mathrm{D}(u,v)$), and u or v is at most 2 hops apart from c and d (since $c \in \mathrm{D}(u,v)$). Hence, x is at most 4 hops apart from c and d in G. Then, there exists a path from u to c and d, which only visits cells in $\mathrm{CN}_2(C(x))$ and thus, $C(c)$ and $C(d)$ are reachable w.r.t. $\mathrm{CN}_2(C(x))$.

Subcase 2.2 ($\measuredangle cud \geq \pi/2$ or $\measuredangle cvd \geq \pi/2$): Assume w.l.o.g. that $\measuredangle cvd \geq \pi/2$ (the other case is symmetric). Then $v \in \mathrm{D}(c,d)$ and $\|vc\|, \|vd\| \leq \mathrm{R}$ holds. The triangle inequality yields

$$\begin{aligned} \|xc\| \quad &\leq \quad \|xv\| + \|vc\| \leq \mathrm{R} + \mathrm{R} = 2\mathrm{R}\,, \text{and} \\ \|xd\| \quad &\leq \quad \|xv\| + \|vd\| \leq \mathrm{R} + \mathrm{R} = 2\mathrm{R}\,. \end{aligned}$$

Hence, $C(c)$ and $C(d)$ belong to $\mathrm{CN}_2(C(x))$. Moreover, it holds that x is at most 2 hops apart from u and v in G (since $x \in \mathrm{D}(u,v)$), and v is at most 2 hops apart from c and d (since $v \in \mathrm{D}(c,d)$). Thus, x is at most 4 hops apart from c and d in G. With the same argument as in Subcase 2.1, $C(c)$ and $C(d)$ are reachable w.r.t. $\mathrm{CN}_2(C(x))$. □

Theorem 8.12. *Let $G = (V,E)$ be a QUDG satisfying $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$ and w be any node in $\mathrm{Virt}(G_b)$. There exists a node $u \in V$, such that cluster head and bridge edge selection in all cells which are connected to u in G by a path of length at most 6 hops and which are reachable w.r.t. $\mathrm{CN}_3(C(u))$, leads to an identical local view of w on the virtual planar backbone graph compared to the construction of $\mathrm{Virt}(G_b)$ over the entire network graph. In case w is a real node then this claim holds for $u = w$.*

Proof. To improve readability, in the following the term *global construction* refers to the construction of $\mathrm{Virt}(G_b)$ over the entire network graph, whereas *local construction* refers to cluster head and bridge edge selection in the restricted area described in this lemma.

Case (w is a real node): By Lemma 8.9 it holds that w is also a backbone node in the local construction starting from $C(w)$.

"⇒": It is first shown that any neighbor v of w in the local construction is also a neighbor of w in the global construction.

Let v by any neighbor of w in the local construction. Backbone node v is either a real node or a virtual node. Suppose it is a real node. All reachable cells w.r.t. $\mathrm{CN}_1(C(v))$ are reachable cells w.r.t. $\mathrm{CN}_2(C(w))$ and are connected to w by at most 3 hops. Hence, by Lemma 8.9 v is a backbone node in the global construction. Now suppose that v

is a virtual node generated by the intersecting backbone edges wu and xy in the local construction. It holds by Lemma 8.8 that the cells $C(u)$, $C(x)$, and $C(y)$ are reachable cells w.r.t. $\mathrm{CN}_2(C(w))$ and are connected to w by at most 3 hops. Hence, the reachable cells w.r.t. these cells' one-cell neighborhoods are at most 5 hops apart from w in G and are reachable cells w.r.t. $\mathrm{CN}_3(C(w))$. Hence, by Lemma 8.9, u, x, y are real and v is a virtual backbone node in the global construction.

Now, in either of the two cases (v is *real* or *virtual*), assume that v is not a neighbor of w in the global construction. Then there must exist a backbone edge ab in the global construction that intersects wv. This leads to a contradiction, because a and b are at most 3 hops apart from w in G and the cells $C(a)$ and $C(b)$ as well as their reachable one-cell neighborhoods are reachable cells w.r.t. $\mathrm{CN}_3(C(w))$, which implies that a and b are also backbone nodes in the local construction and v cannot be a neighbor of w. Hence, v is a neighbor of w in the global construction.

"$\Leftarrow$": It is shown that any neighbor v of w in the global construction is also a neighbor of w in the local construction.

This can be proven analogously to the prior implication, where the terms *local* and *global construction* are simply swapped. The argumentation stays the same.

Case (w is a virtual node): Let x be the generator of w whose existence is guaranteed by Lemma 8.11. Then all real neighbors of w as well as all generators of virtual neighbors of w are 4-hop neighbors of x in G and are contained by reachable cells w.r.t. $\mathrm{CN}_2(C(x))$. For any such cell C it holds that its reachable cells w.r.t. $\mathrm{CN}_1(C)$ are at most 6 hops apart from x and are reachable cells w.r.t. $\mathrm{CN}_3(C(x))$. Consequently, all real neighbors and generators of virtual neighbors of w in the global construction are also backbone nodes in the local construction starting from $C(x)$ and the latter actually contains the virtual node w. Hence, the neighborhoods of w in the local and global construction can only differ if there would be a backbone edge ab in the local construction which intersects backbone edge wv, where v is a virtual or real neighbor of w in the global construction. Suppose such an edge exists. Then using the same argumentation as in the proof of Lemma 8.11 (Cases 1 and 2), it must hold due to the choice of x that a and b are at most 4 hops apart from x in G and that $C(a)$ and $C(b)$ are reachable cells w.r.t. $\mathrm{CN}_2(C(x))$. Hence, their reachable 1-cell neighborhoods are connected by at most 6 hops to x and are reachable cells w.r.t. $\mathrm{CN}_3(C(x))$. But then, a and b are also backbone nodes in the global construction, which is a contradiction to v being a neighbor of w therein. It follows that the neighborhoods of w in the local and the global construction are equivalent, and with x there exists a real node satisfying the claim. □

The above theorem implies correctness of the following algorithmic approach. Let node $u \in V$ be a node that wants to construct its local view on $\mathrm{Virt}(G_b)$. It requests all nodes in G belonging to reachable cells in $\mathrm{CN}_3(C(u))$ by paths of length at most 6 hops in G (e.g. using restricted flooding) to execute the distributed algorithm by Lillis et al. [70, 71]. After termination by all nodes, u knows its adjacency in $\mathrm{Virt}(G_b)$ if it happens to be a backbone node, and it otherwise knows its connecting edge to its cluster head in the routing graph G_r.

In the subsequent sections, it is now shown that this idea can be transformed into a beaconless algorithm.

8.3 Beaconless Computation of Bridge Edges

The first step towards beaconless computation of a nodes adjacency in $\mathrm{Virt}(G_b)$ is the construction of the bridge edges of a particular cell. This is the task of algorithm *BeaconlessBridgeEdge* (BBE) that is introduced now.

Given any QUDG $G = (V, E)$, after execution of BBE by any node $u \in V$, for each cell in $\mathrm{CN}_1(C(u))$ which is connected by at least one edge in E to $C(u)$, node u is aware of exactly one such edge called the *representative bridge edge*. Initially node u is unaware of its neighborhood. During the execution it exchanges messages only with a few neighbors. Apart from worst-case situations, which are considered in Section 8.5, most neighbors of u never send a message and therefore remain invisible for u. This avoids communication overhead compared to any beacon-based approach, where all nodes transmit a message regardless of whether they are important for this task or not.

It is worth noting that algorithm BBE is applicable for arbitrary QUDGs, i.e., it does not require input graphs to satisfy $\mathrm{R/r} \leq \sqrt{2}$.

8.3.1 Description of Algorithm BeaconlessBridgeEdge (BBE)

The algorithm consists of two phases (Phase I and Phase II), which are executed by the initiating node u, and message handling routines for u as well as for all other nodes.

The algorithm is started by the initiator u with execution of Phase I, in which u determines one representative bridge edge to each neighboring cell in $\mathrm{CN}_1(C(u))$ to which it is connected in G. It does so by sending a *FirstRequest* which contains the 1-cell neighborhood list $\mathcal{L}_u$. This FirstRequest initiates a timer-based contention among nodes in its neighboring cells. In all neighboring cells in which this FirstRequest is overheard, nodes overhearing this message start a delay timer proportional to their Euclidean distance to the geometric center of their cell and whose duration is at most $\mathrm{t_{max}}$. As soon as a node's timer expires, it answers with a *FirstResponse*. If a node overhears a FirstResponse from another node in its cell, it refrains from sending a message. This way, after time at most $\mathrm{t_{max}}$ exactly one FirstResponse is generated in each cell that has participated. Initiator u overhears these FirstResponses, stores the respective bridge edges, and removes those cells from $\mathcal{L}_u$. After time $\mathrm{t_{max}}$ it terminates Phase I and starts execution of Phase II.

The goal of Phase II is to determine one representative bridge edge for every neighboring cell in $\mathrm{CN}_1(C(u))$ for which no bridge edge was found during Phase I, i.e., for every cell contained in list $\mathcal{L}_u$. To do so for every such "unprocessed" cell, node u starts a contention among nodes in its cell $C(u)$ by sending a *SecondRequest* containing one cell C from $\mathcal{L}_u$. Upon sending this message, node u starts a delay timer of duration $\mathrm{t_{max}}$. This delay may be extended later on. Those nodes in $C(u)$ whose communication radius R intersect the "unprocessed" cell C become *candidates* for this cell. Candidates start a delay timer of duration at most $\mathrm{t_{max}}$ proportional to their Euclidean distance to the geometric center of C. In turn, the node in $C(u)$ closest to the geometric center of C will be the

first to respond with a Phase II FirstRequest. All candidates as well as initiator u will extend their delay timers by time t_{max}, after overhearing this message. This FirstRequest additionally starts a contention among the nodes in cell C overhearing it, just like in Phase I. Now two cases may occur.

If there is at least one node in C overhearing the message, then the one closest to the geometric center to C will be the first to respond with a Phase II FirstResponse and suppresses other responses from C. Upon overhearing this message, the corresponding candidate sends a SecondResponse. This SecondResponse is overheard by all nodes in $C(u)$. Other candidates for C as well as the initiator u cancel their delay timers. Initiator u stores the respective bridge edge, removes cell C from list $\mathcal{L}_u$, and sends the next SecondRequest unless $\mathcal{L}_u$ is empty.

In the other case, where no node in C has overheard the Phase II FirstRequest by the candidate, after time t_{max} this candidate's delay timer expires without having received a Phase II FirstResponse. In turn, eventually the delay timer of the second closest candidate to C expires (if any), which then performs identical action. Eventually, one of the candidates succeeds in detecting a representative bridge edge—or not. In both cases, initiator u eventually proceeds with the next unprocessed cell, either after having received a SecondResponse, or after expiry of its own delay timer. Initiator u terminates once $\mathcal{L}_u$ is empty and the delay timer of u has expired or canceled.

A detailed pseudocode description is given in Algorithm 3. In this description, variable u always refers to the distinguished node which has initiated the algorithm's execution. Four different message types are used. Any message includes the position of its *sender* s. Cells may, for example, always be represented by their geometric centers. The geometric center (midpoint) of a cell C is denoted by $\mathrm{gc}(C)$.

8.3.2 Correctness

To prove termination, it is shown that every node involved in the computation reaches a state in which no timer is running and no message sending is scheduled.

Theorem 8.13. *Execution of BBE initiated by any node $u \in V$ on a QUDG $G = (V, E)$ terminates.*

Proof. Note that message sending by nodes $v \neq u$, implies expiry, i.e., not cancellation, of delay timers.

After a delay of t_{max} past sending the FirstRequest, initiator u terminates Phase I and every neighbor v of u in G has either decided not to start a timer, has canceled its delay timer, or has actually answered with a FirstResponse; i.e., all of these nodes have terminated at the moment when Phase II is being entered.

At the beginning of Phase II, $|\mathcal{L}_u| \leq |\mathrm{CN}_1(C(u))| \in \mathcal{O}(1)$ and during each loop pass, one element is removed from $\mathcal{L}_u$. Hence, to show termination of Phase II, it suffices to show that a single loop pass terminates.

Let Z_i be the head of list $\mathcal{L}_u$ removed in round i and denote this cell's geometric center by m_i. Let $\mathcal{C}$ be the (possibly empty) distance-ordered set of candidates (which are nodes in $C(u)$) for Z_i, i.e., for any two $c_j, c_l \in \mathcal{C}$, $j < l \Leftrightarrow \|c_j m_i\| < \|c_l m_i\|$.

Algorithm 3 BeaconlessBridgeEdge (BBE)

Variables: u is the initiator, r and R refer to the minimum and maximum transmission radius, respectively, $t_{max} > 0$ is a constant known to all nodes.

Description of message types

FirstRequest(s,$\mathcal{L}_s$,i) contains a list $\mathcal{L}$ of unprocessed cells and integer $i \in \{\text{I,II}\}$ referring to the phase during which the message is sent.

SecondRequest(s,C) contains a single unprocessed cell C.

FirstResponse(s,d,i) is a node's answer to a FirstRequest previously sent by *destination* d which is addressed by this message. Integer $i \in \{\text{I,II}\}$ refers to the phase during which the message is sent.

SecondResponse(s,w,d) is a node's answer to a SecondRequest previously sent by *destination* d which is addressed by this message. Node w is located in the cell addressed by this SecondRequest.

Action of initiator u

1: **Phase** I
2: Set $\mathcal{L}_u \leftarrow \mathrm{CN}_1(C(u)) \setminus \{C(u)\}$ ▷ Initialize list of neighboring cells
3: Send FirstRequest(u,$\mathcal{L}_u$,I) and schedule termination of Phase I after delay of t_{max}
4: Terminate Phase I
5: **Phase** II
6: **while** $\mathcal{L}_u \neq \emptyset$ **do**
7: Remove head of list $\mathcal{L}_u$, denoted by Z
8: Send SecondRequest(u,Z) and start a delay timer of duration t_{max}
9: Terminate Phase II
10: **on** reception of FirstResponse(s,d,i) **do**
11: **if** $i = \text{I}$ **then** ▷ Note that $d = u$ must be the case
12: $\mathcal{L}_u \leftarrow \mathcal{L}_u \setminus \{C(s)\}$ and store us as representative bridge edge for $C(s)$
13: **on** reception of FirstRequest(s,$\mathcal{L}_s$,i) **do**
14: **if** $i = \text{II}$ **then**
15: Extend delay timer by t_{max}
16: **on** reception of SecondResponse(s,w,u) **do**
17: Store sw as representative bridge edge for $C(w)$, cancel delay timer, and continue with the while-loop in line 6

Action of node $v \neq u$

18: **on** reception of FirstRequest(s,$\mathcal{L}_s$,i) **do**
19: **if** $i = \text{I}$ and $C(v) \in \mathcal{L}_u$ **then** ▷ Note that $s = u$, and therefore use variable u instead of s
20: Schedule sending of FirstResponse(v,u,I) after a delay of duration $t(\|v, \mathrm{gc}(C(v))\|, \mathrm{r}/2)$
21: **if** $i = \text{II}$ **then** ▷ Note that $|\mathcal{L}_s| = 1$
22: **if** v is a *candidate* for the cell contained in $\mathcal{L}_s$ **then**
23: Extend delay timer by t_{max}
24: **if** $C(v) \in \mathcal{L}_s$ **then**
25: Schedule sending of FirstResponse(v,s,II) after delay of duration $t(\|v, \mathrm{gc}(C(v))\|, \mathrm{r}/2)$
26: **on** reception of FirstResponse(s,d,i) **do**
27: **if** $C(v) = C(s)$ **then**
28: Cancel delay timer and sending of FirstResponse
29: **if** $i = \text{II}$ and $v = d$ **then**
30: Send SecondResponse(v,s,u)
31: **on** reception of SecondRequest(u,Z) **do**
32: **if** $C(v) = C(u)$ and cell Z intersects v's transmission radius R **then**
33: Become a *candidate* for Z and schedule sending of FirstRequest(v,$\mathcal{L}_v = Z$,II) after delay of duration $t(\|v, \mathrm{gc}(Z)\|, \mathrm{R} + \mathrm{r})$
34: **on** reception of SecondResponse(s,w,u) **do**
35: **if** $C(v) = C(s)$ and v is *candidate* for $C(w)$ **then**
36: Cancel delay timer and sending of FirstRequest

If $\mathcal{C} = \emptyset$, then no node in $C(u)$ schedules sending of a FirstRequest and after time $\mathrm{t}_{\max}$, initiator u proceeds with the next loop pass. Otherwise, let $\mathcal{C}' = \langle c_1, ..., c_g \rangle \subseteq \mathcal{C}$, with $1 \leq g \leq |\mathcal{C}|$, be the longest consecutive subsequence of $\mathcal{C}$, beginning with the first element c_1 in $\mathcal{C}$, s.t. nodes in $\mathcal{C}'$ do not have an edge in E with the second endpoint being contained in cell Z_i.

An inductive argument on the length of $\mathcal{C}'$ yields that after time at most $(g+1) \cdot \mathrm{t}_{\max}$ all nodes in $\{u\} \cup \mathcal{C} \setminus \mathcal{C}'$ have a running delay timer with remaining delay of at most $\mathrm{t}_{\max}$.

If $\mathcal{C} = \mathcal{C}'$, by that time the delay timers of all nodes other than the initiator's have already expired. After a delay of at most $\mathrm{t}_{\max}$, the delay timer of initiator u expires as well; i.e., no node has a running timer or has scheduled sending of a message and the next loop pass begins.

In the remaining case, the next expiring delay timer belongs to c_{g+1}, the first element from $\mathcal{C} \setminus \mathcal{C}'$. By construction of $\mathcal{C}$, $\exists\, c_{g+1}z \in E$, with $z \in Z_i$ and z answers to the FirstRequest by c_{g+1}. This causes c_{g+1} to send a SecondResponse for Z_i, which cancels delay timers and message transmissions of all other candidates in $\mathcal{C}$. Moreover, this leads to cancellation of the delay timer of u, which then proceeds with the next pass of the while-loop. □

To prove correctness, the following notation is required. Denote by $\mathcal{N}(C(u))$ the subset of neighboring cells of $C(u)$, which are connected by a bridge edge to $C(u)$ in the input graph G. Partition $\mathcal{N}(C(u))$ into two disjoint subsets $\mathcal{N}^{(1)}(C(u))$ and $\mathcal{N}^{(2)}(C(u))$ as follows. $\mathcal{N}^{(1)}(C(u))$ encompasses exactly all cells in $\mathrm{CN}_1(C(u))$ to which node u shares an edge in the input graph G, whereas $\mathcal{N}^{(2)}(C(u))$ encompasses the remaining cells from $\mathrm{CN}_1(C(u))$ to which u does not share an edge, but other nodes in $C(u)$ do.

Lemma 8.14. *After termination of Phase I, node u knows exactly one representative bridge edge for each cell reachable cell in* $\mathrm{CN}_1(C(u))$.

Proof. If $\mathcal{N}^{(1)}(C(u)) = \emptyset$, then no neighbor of u in G starts a delay timer for a FirstResponse. After time $\mathrm{t}_{\max}$, u terminates with an empty set of representative bridge edges.

Otherwise, let C be any cell in $\mathcal{N}^{(1)}(C(u))$. By definition of this set, there exists $V' \subset V$, $V' \neq \emptyset$, whose elements are contained by C and share an edge with u in E. The node in V' closest to the geometric center of C is the first to send a FirstResponse. This message is overheard by u, which stores uv as the representative bridge edge, and it is overheard by all other nodes in V', which causes cancellation of these nodes' timers and scheduled transmissions. □

Lemma 8.15. *After termination of Phase II, node u knows exactly one representative bridge edge for each cell in $\mathcal{N}^{(2)}(C(u))$.*

Proof. After termination of Phase I, list $\mathcal{L}_u$ is a superset of $\mathcal{N}^{(2)}(C(u))$. Before the first execution of the while-loop, $0 < |\mathcal{L}_u| \leq |\mathrm{CN}_1(C(u))|$ holds. Hence, the loop is actually being executed. As the size of $\mathcal{L}_u$ decreases in every round by one element, there are at most $|\mathrm{CN}_1(C(u))| \in \mathcal{O}(1)$ rounds of execution.

For an arbitrary $i \leq |\mathrm{CN}_1(C(u))|$ it is shown: If cell C_i which is processed during round i belongs to $\mathcal{N}^{(2)}(C(u))$, then at the end of round i, u stores a representative bridge

edge connecting $C(u)$ and C_i. This proves the claim since the execution of a round is independent of the outcome of previous rounds. Distinguish two cases:

Case ($C_i \in \mathcal{N}^{(2)}(C(u))$): There is a non-empty subset $V' \subset V$ of candidates for C_i, all of which are connected in G to at least one node in C_i. Let $v \in V'$ be the node minimizing the distance to the geometric center of C_i. At the moment this node's delay timer expires, it has not overheard any other SecondResponse for C_i. This follows from the choice of v. Hence, v sends a FirstRequest for C_i, which is being answered by some neighbor $w \in C_i$ whose existence is guaranteed by assumption. In consequence, v sends a SecondResponse. This is overheard by all nodes in $C(u)$. Node u stores vw as bridge edge for C_i and other candidates in V' cancel their delay timers and scheduled transmissions.

Case ($C_i \notin \mathcal{N}^{(2)}(C(u))$): There is either no candidate in $C(u)$, or none of the candidates shares an edge in G with an endpoint in C_i. In the former case, no message transmissions are scheduled by nodes in $C(u)$. After delay $\mathrm{t}_{\max}$, u continues with the next loop pass. In the other case, each candidate sends a FirstRequest for C_i, but which is not being answered. Hence, no candidate sends a SecondResponse and u continues with the next loop pass. In either case, u proceeds without storing a representative edge for C_i. □

Theorem 8.16. *After the execution of algorithm BBE on any QUDG $G = (V, E)$ by any $u \in V$, u stores one bridge edge for each cell in $\mathcal{N}(C(u))$.*

Proof. The proof follows directly from Lemma 8.14 and Lemma 8.15. □

8.4 Beaconless Construction of the Virtual Planar Backbone Graph

The desired goal is to design a reactive local topology control algorithm for construction of a node's local view on the virtual planar backbone graph. Given a distinguished node $u \in V$, by Theorem 8.12 it is sufficient to select cluster heads and bridge edges in all reachable cells w.r.t. $\mathrm{CN}_3(C(u))$ which are connected to $C(u)$ by at most 6 hops. Algorithm BBE from the previous section enables beaconless selection of bridge edges in a single cell. In the following this algorithm is extended to algorithm *BeaconlessBackbone* (BBB). Executed by an arbitrary node u it first selects a cluster head for $C(u)$. This cluster head then constructs bridge edges to neighboring cells using the technique from BBE. In turn any neighboring cell of $C(u)$ to which a bridge edge is constructed also starts algorithm BBB, as long as it belongs to $\mathrm{CN}_3(C(u))$ and is connected to $C(u)$ by at most 6 hops, which again causes execution of BBB in its neighboring cells, and so on.

Algorithm BBB ensures that after its termination, which is detected by the cluster head of the distinguished cell $C(u)$, cluster head and bridge edge selections have been successfully performed by all relevant cells. Then, each node in each of these cells knows whether or not it belongs to the backbone. Then, the induced subgraph over these backbone nodes is constructed and each backbone node in $C(u)$ gathers 3-hop neighborhood information in this graph. This information is sufficient for each of these backbone nodes to compute its adjacency in $\mathrm{Virt}(G_b)$. In the following the algorithm is developed step-by-step.

8.4.1 Beaconless Cluster Head Selection

Let $x \in V$ be any node and let $\mathrm{gc}(C(x))$ denote the geometric center of $C(x)$. Beaconless selection of a cluster head in $C(x)$, initiated by x, can be achieved as follows.

Node x locally broadcasts a message of type *CH-Request* including its position. Upon reception of this message, every node $v \in V$ with $v \in C(x)$ (including x), schedules sending of a message of type *CH-Response*, including its position, after delay of $t(\|v, \mathrm{gc}(C(x))\|, \mathrm{r}/2)$. If upon expiry of its delay timer, node v has not overheard a CH-Response by any other node $w \in C(x)$, then it locally broadcasts the CH-Response. Thereby, the sender of the CH-Response announces itself as cluster head of $C(x)$, denoted $\mathcal{H}(C(x))$. All other nodes in $C(x)$ overhear this message, cancel their delay timers and scheduled transmissions, and store v as cluster head.

Note that this algorithm is correct. There is at least one node, namely initiator x, contained by the corresponding cell. Hence, one node actually responds and suppresses all other CH-Responses.

8.4.2 Beaconless Backbone Construction

The combination of beaconless cluster head selection and beaconless bridge edge construction provides the basic functionalities of the desired algorithm.

However, the broadcast-like operation outlined above misses one important feature, namely, *termination detection*. Although the cluster head of the distinguished cell $C(u)$ is aware of the termination of bridge edge selection process in its cell $C(u)$, it cannot possibly know when this operation has terminated in all other relevant cells in $\mathrm{CN}_3(C(u))$. This is due to the following fact: During Phase II of bridge edge selection, for construction of a representative bridge edge to a particular C it may be the case that not only one, but multiple, or even all candidates (which can be arbitrarily many in the worst-case) have to try to find a neighbor in cell C. Unless assumptions are made that allow to upper-bound the time required to process a single neighboring cell (e.g., by making assumptions on the maximum number of nodes in the network, or the like), the time required till termination of beaconless bridge edge selection cannot be known or bounded in advance.

The solution to this problem in algorithm BBB is a *convergecast* operation. The broadcast-like execution of cluster head and bridge edge selection gives a tree T of cells, rooted at distinguished cell $C(u)$. In this tree, cell C is the parent of cell C' if bridge edge selection in C triggered algorithm execution in C'. Leaves in this tree are those cells that have not triggered algorithm execution in any other cell. Once beaconless bridge edge selection by the cluster head of such a leaf cell has terminated, this cluster head can report its termination to its parent in T. Once an internal node in T has terminated bridge edge selection and all children in T have reported their termination, it can report termination to its parent, and so on. In algorithm BBB, this operation is performed by the cluster head of a cell, which maintains a list $\mathcal{L}_{\mathrm{trig}}$ of triggered neighboring cells.

Next, algorithm BeaconlessBackbone (BBB) is described as a whole.

Algorithm BeaconlessBackbone (BBB) is started on input $C(u)$ by a single node $u \in V$ by sending a CH-Request. As described previously, this message starts a contention among

all nodes in $C(u)$ and the node closest to the geometric center of $C(u)$ announces itself as cluster head $\mathcal{H}(C(u))$. This cluster head now starts beaconless bridge edge construction as described in Algorithm 3. Each node v sending a FirstResponse during bridge edge construction, which is outside of $C(u)$, inside $\mathrm{CN}_3(C(u))$, and which is at most 6 hops apart from $C(u)$, starts execution of algorithm BBB on input $C(u)$, unless this has already been done by another node in its cell $C(v)$. Moreover, if node v executes BBB, then its sends a message of type *Triggered* to the destination node of its FirstResponse. If this destination node is not the cluster head of its own cell, then it forwards this message to its cluster head. Thus, in either case this cluster head is aware of the fact that cell $C(v)$ has been triggered by its bridge edge construction and stores this in $\mathcal{L}_{\mathrm{trig}}$, the list of triggered neighboring cells.

After termination of Phase II of beaconless bridge edge construction by any cluster head $\mathcal{H}(C)$, this cluster head has either an empty list $\mathcal{L}_{\mathrm{trig}}$—or not.

- If this list is not empty, then the cluster head waits for incoming messages of type *Terminated* from triggered neighboring cells. Upon receiving such a message, it removes the corresponding cell from its list $\mathcal{L}_{\mathrm{trig}}$.
- If this list is empty and it is not the cluster head of distinguished cell $C(u)$, then its sends a message of type *Terminated* to the node in its cell which has initiated execution of BBB in its cell by sending the CH-Response. This node then forwards the message to the node in the neighboring cell by which it was triggered. If this node is not the cluster head, one additional forwarding step is required such that the corresponding cluster head finally receives this Terminated message.
- If this list is empty and it is cluster head of distinguished cell $C(u)$, then it informs all nodes in $C(u)$ about the termination via a local broadcast and terminates.

The above algorithm is given by the combination of Algorithms 3 and 4, where the latter extends the former by adding two operations[5] and several message handlers. These extensions require that the messages of the types FirstRequest, FirstResponse, SecondRequest, and SecondResponse (introduced in Algorithm 3) as well as the message types introduced in Algorithm 4 additionally contain information on distinguished cell $C(u)$. Moreover, the message types FirstRequest and SecondRequest have to be extended by an integer variable *hopCount*, that counts the number of hops, by which a cell is connected to $\mathcal{H}(C(u))$. Then, each node v receiving such a message can immediately decide if its cell $C(v)$ is at most 6 hops apart from $C(u)$ and if in addition $C(v) \in \mathrm{CN}_3(C(u))$ holds.

[5] *After sending a FirstResponse* has to be executed by a node directly after sending a FirstResponse in Algorithm 3.
After termination of Phase II is executed by every cluster head that has just terminated bridge edge selection, i.e., that has reached Step 9. in Algorithm 3.

Algorithm 4 BeaconlessBackbone (BBB)

Assumes: inital call by distinguished node u on input $C(u)$.

Variables: r and R refer to the minimum and maximum transmission radius, respectively, $t_{max} > 0$ is a constant known to all nodes.

Description of message types

CH-Request(s) requests election of a cluster head in cell $C(s)$ by sender s.

CH-Response(s) is the announcement of sender s that it becomes cluster head of cell $C(s)$.

Triggered(s,d,C) informs destination d that its FirstRequest triggered algorithm execution in cell C.

Terminated(s,d,C) informs destination d that algorithm execution in cell C has terminated.

Action of node v

1: Send CH-Request(v) and enqueue this message also in own inbox

Action of node v after sending FirstResponse(v, d, i) ▷ Extension to Algorithm 3

2: **if** $C(v) \in \mathrm{CN}_3(C(u))$, v is connected to $C(u)$ by at most 6 hops, and as yet no CH-Request for $C(v)$ has been processed **then**
3: *triggeredBy* $\leftarrow d$, send Triggered$(v, d, C(v))$, and execute BBB$(C(u))$

Action of node v after Termination of Phase II ▷ Extension to Algorithm 3

4: **while** $\mathcal{L}_{\mathrm{trig}}(v) \neq \emptyset$ **do** ▷ There are triggered neighboring cells that have not terminated yet
5: Wait for messages of type Terminated and process those
6: **if** $C(v) \neq C(u)$ **then** ▷ Node v is not cluster head of distinguished cell $C(u)$
7: Send Terminated$(v, y, C(v))$ to $y \in C(v)$, where y is the node having sent the CH-Request for $C(v)$ (if $y = v$, then enqueue this message in own inbox)
8: **else** ▷ Node v is the cluster head of distinguished cell $C(u)$
9: Terminate (if needed, inform nodes in $C(u)$ about termination via a local broadcast)

Message handling by any node v

10: **on** reception of CH-Request(s) **do**
11: **if** $C(v) = C(s)$ **then**
12: Schedule sending of CH-Response(v) and construction of representative bridge edges (as described in Algorithm 3) after delay of duration $t(\|v, \mathrm{gc}(C(v))\|, \mathrm{r}/2)$
13: **on** reception of CH-Response(s) **do**
14: **if** $C(s) = C(v)$ and $s \neq v$ **then**
15: Cancel delay timer, sending of CH-Response, and construction of representative bridge edges
16: Store $\mathcal{H}(C(v)) \leftarrow s$
17: **on** reception of Triggered(s, d, C) **do**
18: **if** $d = v$ and $v \neq \mathcal{H}(C(v))$ **then**
19: Send Triggered$(v, \mathcal{H}(C(v)), C)$ ▷ Forward to cluster head
20: **if** $d = v$ and $v = \mathcal{H}(C(v))$ **then**
21: $\mathcal{L}_{\mathrm{trig}}(v) \leftarrow \mathcal{L}_{\mathrm{trig}}(v) \cup \{C\}$ ▷ Update list of triggered neighboring cells
22: **on** reception of Terminated(s, d, C) **do**
23: **if** $v = d$ **then**
24: **if** $C(v) = C$ **then** ▷ I.e., v has sent the CH-Request for $C(v)$
25: Send Terminated$(v, \mathit{triggeredBy}, C)$ ▷ Forward to parent
26: **if** $v \neq \mathcal{H}(C(v))$ **then**
27: Send Terminated$(v, \mathcal{H}(C(v)), C)$ ▷ Forward to cluster head
28: **if** $v = \mathcal{H}(C(v))$ **then**
29: $\mathcal{L}_{\mathrm{trig}}(v) \leftarrow \mathcal{L}_{\mathrm{trig}}(v) \setminus \{C\}$ ▷ Remove C from list of triggered neighboring cells

8.4.3 Correctness of Beaconless Backbone Construction

Lemma 8.17. *The cluster head of the distinguished cell $C(u)$ terminates eventually.*

Proof. Execution of BBB in $C(u)$ triggers execution of it in neighboring cells, as long as they are at most 6 hops apart from $C(u)$ and belong to $\mathrm{CN}_3(C(u))$. This gives a directed tree of cells, rooted at $C(u)$, containing an edge $\overrightarrow{CC'}$ if cell C has triggered cell C'. Each cell has only a constant number of neighboring cells and the number of cells contained in $\mathrm{CN}_3(C(u))$ is only a constant factor larger. Each cell can only be triggered once. Hence, T consists of a constant number of internal cells and leaf cells.

Leaf cells represent those cells which have not triggered others. Hence, their cluster heads have empty lists of triggered neighboring cells, $\mathcal{L}_{\mathrm{trig}}$. Eventually, these cluster heads have processed all neighboring cells and terminate Phase II, by Theorem 8.13, and then send a message of type Terminated. Cluster heads of parent cells of these leaf cells in T receive and process the Terminated messages and remove the corresponding cells from their lists of triggered neighboring cells. An induction on the height of tree T yields that, whenever all cells of height i have terminated, then all cluster heads of cells of height $i-1$ have empty lists of triggered neighboring cells and terminate as well. Therefore, eventually all successors of the root node have terminated and $\mathcal{H}(C(u))$ receives terminated messages from all such cells. Then $\mathcal{H}(C(u)))$ has an empty list of triggered neighboring cells and it terminates. □

Lemma 8.18. *Once the cluster head of distinguished cell $C(u)$ has terminated, algorithm BBB has been executed in every reachable cell from $\mathrm{CN}_3(C(u))$ which is at most 6 hops apart from $C(u)$.*

Proof. Assume there is $Z \in \mathrm{CN}_3(C(u))$, which is reachable w.r.t. $\mathrm{CN}_3(C(u))$ and connected to $C(u)$ by a path of length at most 6 hops, but in which BBB has not been executed. Because Z is reachable, there exists a path $\Pi_G(x,z)$, with $x \in C(u)$ and $z \in Z$, connecting x and z in G, s.t. all nodes of this path belong to $\mathrm{CN}_3(C(u))$. Let $\mathcal{C} = \langle C(u) = C_1, ..., C_l = Z \rangle$, $l \geq 2$, be the sequence of cells visited, while traversing $\Pi_G(x,z)$. The claim is proven by induction on the length of $\mathcal{C}$.
I.h.: The claim holds for an arbitrary but fixed i.
B.c. ($|\mathcal{C}| = 2$): By Theorem 8.16, during computation of representative bridge edges, one such edge c_1c_2 connecting $c_1 \in C_1$ and $c_2 \in C_2$ is found. At this point, either another node in C_2 has already started execution of BBB, or c_2 does so.
I.s. ($i \to i+1$): By the i.h., it holds that all cells C_i, $i < |\mathcal{C}|$, have executed BBB. By the time it is executed in C_i, C_{i+1} has either already been triggered by some other cell in $\mathrm{CN}_3(C(u))$, or not. Consider the latter case. By assumption there exists at least one edge $c_ic_{i+1} \in \Pi_G(x,z)$ with $c_i \in C_i$ and $c_{i+1} \in C_{i+1}$. By Theorem 8.16 one such edge is eventually determined. By assumption, C_{i+1} is reachable w.r.t. $\mathrm{CN}_3(C(u))$ and connected to $C(u)$ by a path of length at most 6 hops. Hence, the corresponding endpoint in C_{i+1} executes BBB. □

Theorem 8.19. *After termination of algorithm BBB by the cluster head of distinguished cell $C(u)$, in every reachable cell from $\mathrm{CN}_3(C(u))$ which is at most 6 hops apart from*

$C(u)$ the cluster head and all bridge edges have been selected and each node in these cells knows its role in the backbone graph.

Proof. The execution of BBB by any node v includes the selection of a cluster head and the computation of the set of representative bridge edges by that cluster head for cell $C(v)$. By Lemma 8.18, these operations are performed by all relevant cells from $\mathrm{CN}_3(C(u))$. Hence, after termination at $\mathcal{H}(C(u))$ each node in each of these cells knows whether or not it belongs to the backbone. □

8.4.4 Construction of Nodes' Adjacencies in the Virtual Planar Backbone

The final task is to equip all backbone nodes that are residing in distinguished cell $C(u)$ with their adjacencies in the virtual planar backbone graph $\mathrm{Virt}(G_b)$.

In the following assume algorithm BBB has been executed by a node $u \in V$ of a given QUDG $G = (V, E)$ and cluster head $\mathcal{H}(C(u))$ has reported the algorithm's termination. Let $v \in C(u)$ be any backbone node. By Lemma 8.8 it holds that only those backbone edges incident to backbone nodes residing in reachable cells w.r.t. $\mathrm{CN}_2(C(u))$ can intersect backbone edges incident to v. Moreover, by Theorem 8.12, all such backbone nodes have actually been constructed. Let $V_b' \subset V_b$ be the subset of backbone nodes residing in cells that are reachable w.r.t. $\mathrm{CN}_2(C(u))$. The induced graph G_b' w.r.t. input graph G, i.e., the graph over node set V_b' where any two nodes are connected by an edge if they are connected in E, is a connected quasi unit disk graph satisfying $\mathrm{R/r} \leq \sqrt{2}$. This holds because G_b' is connected. The latter holds for the following reason: Firstly, BBB was executed in any cell $C \in \mathrm{CN}_3(C(u))$ by a node which is connected by a path, consisting of nodes that belong to V_b', to the distinguished cell $C(u)$. Secondly, all nodes that belong to C are connected by a clique in G. Hence, the nodes in V_b' constitute a connected set in G. In addition, the graph induced over a subset of nodes from a QUDG $G = (V, E)$ satisfying $\mathrm{R/r} \leq \sqrt{2}$ is also a QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$. By Lemma 3.12, it now follows that each real backbone node v in $C(u)$ can compute its adjacency in $\mathrm{Virt}(G_b)$ if it is provided with 3-hop neighborhood information of G_b'.

Note that gathering 3-hop neighborhood information by a node in G_b' does not contradict the reactive approach, since the degree of any node in G_b' is bounded from above by a constant and therefore, the size of its 3-hop neighborhood is in $\mathcal{O}(1)$ as well (i.e, the costs produced by gathering 3-hop neighborhood information is asymptotically the same as sending of a single message).

Clearly, gathering of k-hop neighborhood information is easy and straightforward. However, for completeness of the approach, now a simple algorithm called BackboneNeighborhood (BBN) is presented which completes this task. If executed by the cluster head $\mathcal{H}(C(u))$ of distinguished cell $C(u)$ right after termination of BBB, it provides all backbone nodes in $C(u)$ with their k-hop neighborhood in G_b'.

In the remainder, let the k-hop neighborhood of a node $v \in V$ in $G = (V, E)$ be defined as follows: $\mathrm{N}_0^G(v) = \{v\}$ and for $k \geq 1$, $\mathrm{N}_k^G(v)$ is the subset of nodes from V reachable by v in G along a path consisting of at most k hops.

Algorithm BBN executed by $\mathcal{H}(C(u))$ on input k proceeds as follows.

In rounds $i = 0..k-1$, $\mathcal{H}(C(u))$ locally broadcasts a message of type *Neighborhood*, including number i as well as its i-hop neighborhood information, and delays the next round by using a delay timer of duration $(k+1-i) \cdot \mathrm{t_{max}}$. This delay ensures that all backbone nodes which are at most $(k-i)$ hops apart from $\mathcal{H}(C(u))$ have learned about their complete $(i+1)$-hop neighborhoods before $\mathcal{H}(C(u))$ sends the next message.

Upon reception of such a message, any backbone node v (including $\mathcal{H}(C(u))$) which belongs to $\mathrm{CN}_2(C(u))$ updates its $(i+1)$-neighborhood information accordingly. Any node $v \neq \mathcal{H}(C(u))$, which receives a Neighborhood message for number i for the very first time, schedules transmission of a Neighborhood message containing number i and its i-hop neighborhood information after a delay of at most $\mathrm{t_{max}}$ and proportionally to the Euclidean distance to sender of the Neighborhood message.

Upon termination by $\mathcal{H}(C(u))$, any backbone node in $C(u)$ is then provided with its full k-hop neighborhood information.

A detailed pseudocode description of algorithm BBN is given in Algorithm 5.

Algorithm 5 BackboneNeighborhood (BBN)

Assumes: execution of algorithm BBB on input $C(u)$ by node u has terminated.

Variables: u is the distinguished node, r and R refer to the minimum and maximum transmission radius, respectively, $\mathrm{t_{max}} > 0$ is a constant known to all nodes.

Definition of message types

Neighborhood(s,$C(u)$,i,$\mathrm{N}_i^{G'_b}(s)$**)** contains the positions of sender s, distinguished cell $C(u)$, an integer $0 \leq i < k$, and the i-hop neighborhood set of s in G'_b.

Action by node $v = \mathcal{H}(C(u))$

1: **for** $i = 0$ to $k-1$ **do**
2: Send Neighborhood(v,$C(u)$,i,$\mathrm{N}_i^{G'_b}(v)$)
3: Start a delay timer with delay of duration $(k+1-i) \cdot \mathrm{t_{max}}$ for delaying the next loop pass
4: Terminate
5: **on** reception of Neighborhood(s,$C(u)$,i,$\mathrm{N}_i^{G'_b}(s)$) **do**
6: $\mathrm{N}_{i+1}^{G'_b}(v) \leftarrow \mathrm{N}_{i+1}^{G'_b}(v) \cup \mathrm{N}_i^{G'_b}(s)$

Action by any node $v \neq \mathcal{H}(C(u))$

7: **on** reception of Neighborhood(s,$C(u)$,i,$\mathrm{N}_i^{G'_b}(s)$) **do**
8: **if** $C(v) \in \mathrm{CN}_2(C(u))$ and v is a backbone node (i.e., $v \in V'_b$) **then**
9: $\mathrm{N}_{i+1}^{G'_b}(v) \leftarrow \mathrm{N}_{i+1}^{G'_b}(v) \cup \mathrm{N}_i^{G'_b}(s)$
10: Unless sending of Neighborhood(v,$C(u)$,i,$\mathrm{N}_i^{G'_b}(v)$) has already been scheduled, schedule sending of it after delay of duration $t(\|vs\|, \mathrm{R})$

Theorem 8.20. *After termination of BBN on input k, initiated by $\mathcal{H}(C(u))$, each backbone node in $C(u)$ knows all its k-hop neighbors in G'_b.*

Proof. By induction on control variable i it is proven that after round i of the loop, all nodes in $\mathrm{N}_{k-i}^{G'_b}(\mathcal{H}(C(u)))$ know their complete $(i+1)$-hop neighborhoods in G'_b.

I.h.: The claim holds for an arbitrary but fixed $i < k - 1$.
B.c. ($i = 0$): After round $i = 0$, i.e., after time $(k + 1) \cdot \mathrm{t}_{\max}$, each $v \in \mathrm{N}_{k+1}^{G'_b}(\mathcal{H}(C(u)))$ has locally broadcasted its set $\mathrm{N}_0^{G'_b}(v)$. Hence, all nodes in $\mathrm{N}_{k-i=k}^{G'_b}(\mathcal{H}(C(u)))$ know their complete $i + 1 = 1$-hop neighborhoods in G'_b.
I.s. ($i \to i + 1$): By the induction hypothesis, after round i all nodes in $\mathrm{N}_{k-i}^{G'_b}(\mathcal{H}(C(u)))$ have complete $(i + 1)$-hop neighborhood information. During iteration $i + 1$, all nodes in $\mathrm{N}_{k+1-(i+1)=k-i}^{G'_b}(\mathcal{H}(C(u)))$ locally broadcast their $(i + 1)$-hop neighborhoods. Hence, after round $i + 1$, all nodes in $\mathrm{N}_{k-i-1}^{G'_b}(\mathcal{H}(C(u)))$ have complete $(i + 2)$-hop neighborhood information in G'_b. Thus, after round $k - 1$, all nodes in $\mathrm{N}_{k-(k-1)=1}^{G'_b}(\mathcal{H}(C(u)))$ have complete $k - 1 + 1 = k$-hop information. Since all backbone nodes in $C(u)$ are 1-hop neighbors of $\mathcal{H}(C(u))$, this proves the claim. □

8.4.5 Summary

Given any node v from a restricted QUDG, this node can compute its role in the virtual planar backbone graph by executing algorithm BeaconlessBackbone (BBB) on input $C(v)$. If it is a non-backbone node, then it instantly knows its adjacency in the routing graph G_r, since in this graph it is simply connected to its cluster head $\mathcal{H}(C(v))$. Otherwise, if it is a backbone node and it requires its adjacency in $\mathrm{Virt}(G_b)$ and G_r, respectively, then it either executes algorithm BackboneNeighborhood (BBN) on input $k = 3$, or asks its cluster head $\mathcal{H}(C(v))$ to do so. Given this information it can compute its adjacency in $\mathrm{Virt}(G_b)$ without any additional communication. In the following analysis and discussion, the combination of algorithms BBB and BBN is referred to as *ReactiveBackbone*.

8.5 Algorithm Analysis

Next, worst-case message complexity of ReactiveBackbone, as well as the size of messages used therein, is analyzed. Afterwards, it is shown that this approach is an Ω-reactive topology control algorithm.

Theorem 8.21 (Message complexity and size)**.** *Let $G = (V, E)$ be a QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$ with $|V| = n$. In the worst-case, the total number of messages transmitted during execution of ReactiveBackbone is $\Theta(n)$. Under the assumption that nodes do not have any a priori neighborhood knowledge, this message complexity is worst-case optimal up to constant factors. Moreover, each message is of size at most $\mathcal{O}(\mathrm{P}_{\max})$ bits, where $\mathrm{P}_{\max}$ is the number of bits required to represent a single geographic position.*

Proof. While assuming that $\mathrm{R/r} \leq \sqrt{2}$, it holds that $|\mathrm{CN}_k(C)| \in \mathcal{O}(1)$ for some constant $k \geq 1$. By Lemma 4 in [71] it holds that the number of backbone nodes in $\mathrm{CN}_k(C)$, for constant k, is $\mathcal{O}(1)$.

During the execution of algorithm BBN on input $k = 3$, each of the $\mathcal{O}(1)$ backbone nodes in reachable cells in $\mathrm{CN}_2(C(u))$ transmits exactly three messages. Hence this overhead is constant.

Cluster head selection requires two message transmissions per cell. Moreover, in each cell at most one message of the types Triggered and Terminated are produced, respectively. Any such message is forwarded at most a constant number of times. Hence, the total number of messages used for cluster head selection, termination detection, and gathering of backbone neighborhood information is bounded by a constant.

By far the most expensive part w.r.t. message complexity is the construction of representative bridge edges. Consider any cell C. The number of FirstRequest/FirstResponse messages (during Phase I) as well as the number of SecondRequests (during Phase II) is upper bounded by $|\mathrm{CN}_1(C)|+1$. During Phase II any node in C, which may be as much as $\mathcal{O}(n)$ in the worst-case, transmits at most $|\mathrm{CN}_1(C)|$ FirstRequests. These FirstRequests generate at most $|\mathrm{CN}_1(C)|$ FirstResponses from nodes in neighboring cells, succeeded by at most $|\mathrm{CN}_1(C)|$ SecondResponse messages. Hence, the worst-case message complexity per cell is dominated by sending of up to $\mathcal{O}(n)$ FirstRequests during Phase II. Then the worst-case message complexity over all cells is $\mathcal{O}(n)$.

To see that the construction of beaconless bridge edges requires at least $\Omega(n)$ message transmissions, recall the example graph in Figure 5.2 used for the proof of Theorem 5.9. During algorithm execution initiated by any $v_i \in V$, every single node in V eventually sends one FirstRequest for C and hence, $\Omega(n)$ messages are transmitted.

Note that unless nodes are provided with a priori neighborhood knowledge, the latter argument implies that the algorithm at hand is worst-case optimal up to constant factors regarding message complexity.

Finally, any message transmitted during algorithm execution contains at most a constant number of geographic positions plus a constant number of additional control bits. Therefore, each message is of size at most $\mathcal{O}(\mathrm{P}_{\max})$ bits, where $\mathrm{P}_{\max}$ denotes the number of bits needed to represent a geographic position. □

Theorem 8.22. *Algorithm ReactiveBackbone is an Ω-reactive local view topology control algorithm for the k-local topology mapping described by Lillis et al. [70, 71].*

Proof. According to Theorem 8.4, the topology mapping described by Lillis et al. [70, 71] is a k-local topology mapping for $k = 8$. By correctness of algorithms BBB (Theorem 8.19) and BBN (Theorem 8.20), as well as Theorem 8.12 it holds that for any node $v \in \mathrm{Virt}(G_b)$ there exists a node $u \in G$, such that ReactiveBackbone executed by u correctly computes v's local view on $\mathrm{Virt}(G_b)$, and hence, it is a local view topology control algorithm. It remains to show that ReactiveBackbone is an Ω-reactive algorithm according to Definition 5.5.

Note that prior to algorithm execution, nodes are not provided with neighborhood knowledge and that it is started by a single local broadcast. Next, consider its message complexity w.r.t. the number of transmitted bits. By Theorem 5.9, the set of representative bridge edges for a cell C cannot be computed by any $\mathcal{O}$-reactive algorithm, since in the worst-case situation depicted in Figure 5.2 any node in C, which may be arbitrarily many, has to transmit at least one message. Hence, algorithm ReactiveBackbone is not $\mathcal{O}$-reactive. Now, it remains to show that $\Omega(|G[u,k]| \cdot \log n)$ is not a lower bound for the message complexity in terms of number of transmitted bits.

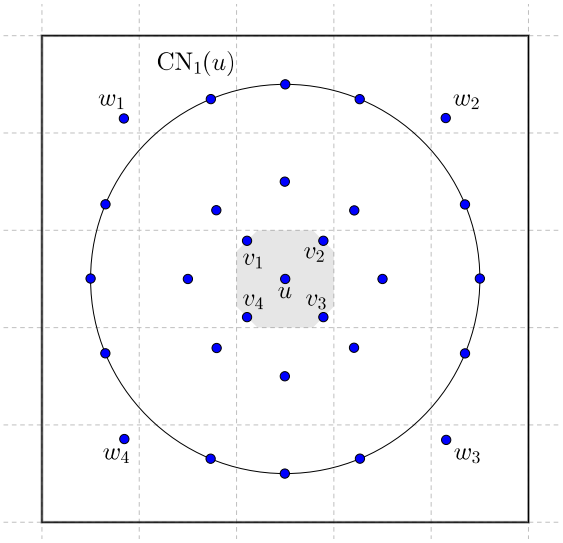


Figure 8.4: Example used in the proof of Theorem 8.22. Only a constant fraction of V is shown, all remaining nodes are contained in the shaded area in $C(u)$.

Consider the restricted QUDG in Figure 8.4, where only a constant fraction $m' \in \mathcal{O}(1)$ of nodes from V is depicted. The remaining $n - m' = m \in \mathcal{O}(n)$ nodes $\{x_1, ..., x_m\}$ are located in the shaded area of cell $C(u)$ and are not shown to keep the figure clear. Node u is connected to any node within its transmission radius R, given by the circle. In addition, each of the nodes v_i is connected to w_i, for $i = 1..4$. All other nodes can be connected arbitrarily as long as their edges obey the QUDG properties. The shaded area represents cell $C(u)$ without the four circles centered at w_i and radius $\|w_i v_i\|$. This graph is connected and it holds that all nodes from V are located in $\mathrm{CN}_1(C(u))$. In particular, it holds that $|G[u, k]| = n$.

Now consider the execution of ReactiveBackbone by node u. During Phase I of bridge edge construction, node u determines one bridge edge for each cell in $\mathrm{CN}_1(C(u))$, except for the four cells containing the nodes w_i. During Phase II, each of the nodes v_i becomes candidate for cell $C(w_i)$ and successfully determines the bridge edge $v_i w_i$. In total, these operations produce a constant number of message transmissions and none of the nodes x_i transmitted any message.

In each of the neighboring cells C_i of $C(u)$, the single node contained in that cell also starts executing BBB. Since these cells contain only one node, the total number of messages generated per cell is bounded by a constant. Note that during these executions at most $|\mathrm{CN}_1(C(u))|$ many nodes x_i may also be forced to transmit a message.

After termination at node u, at most $m' + |\mathrm{CN}_1(C(u))| \in \mathcal{O}(1)$ many nodes have actively participated in algorithm execution by sending a message. Furthermore, each such node has transmitted at most a constant number of messages. During subsequent execution of BBN on input $k = 3$, each backbone node also transmits a constant number of messages. Since there are only $\mathcal{O}(1)$ backbone nodes, this overhead is constant in total. Therefore, the overall number of messages transmitted during execution of ReactiveBackbone is

$\mathcal{O}(1)$. Hence, $\Omega(|G[u,k]) \cdot \log n)$ is not a lower bound for the message complexity in terms of number of transmitted bits. □

For simplicity of the proof, the graph used above is rather regular. However, it is important to note that the beneficiality of algorithm ReactiveBackbone is not limited to such artificial graphs, but can be observed in arbitrary graphs, as long as they do not coincide with worst-case situations as given in Figure 5.2.

8.6 Reactive Geographic Routing on Topological Quasi Unit Disk Spanners

The last contribution of this chapter is a short overview on how algorithm Reactive-Backbone, in particular its constituting routines BBB and BBN, can be used to derive a reactive local geographic routing protocol that guarantees delivery on connected QUDGs satisfying $\mathrm{R/r} \leq \sqrt{2}$.

Reactive-Virtual-Face-Traversal, described next, is similar to the routing protocol *Route1* described by Lillis et al. in [70, 71] and mainly extends the Virtual-Face-Traversal protocol by Guan et al. [72, 73] (see Section 3.2 for details).

Given a restricted QUDG $G = (V, E)$, suppose some source node $s \in V$ has to route a packet to a destination $d \in V$, whose location is known in advance. If s and d are at distance at most r from each other, then s forwards the packet directly to d. Otherwise, s executes algorithm BBB on input $C(s)$. After cluster head $\mathcal{H}(C(s))$ has reported termination of execution, node s knows whether it belongs to the backbone, or not. If not, then it forwards the packet to its cluster head $\mathcal{H}(C(s))$. In either case, the packet is now held by a backbone node in $C(s)$. This backbone node now executes algorithm BBN on input $k = 2$, which provides this node with 2-hop neighborhood information on a connected QUDG satisfying $\mathrm{R/r} \leq \sqrt{2}$. Now *Virtual-Face-Traversal* is applied. The latter updates the packet header according to the virtual routing path and computes a real path consisting only of backbone nodes. The backbone node forwards the packet according to the real routing path. While the packet has not been received by a backbone node which resides in the same cell as the destination node d, the backbone node v currently holding the packet, first executes algorithm BBB on input $C(v)$, waits for termination announced by its cluster head, executes BBN on input $k = 2$, and applies *Virtual-Face-Traversal*. Eventually, the packet will be received either by the destination itself, or a backbone node residing in the same cell as d. In the latter case, this backbone node forwards the packet to d.

Theorem 8.23. *Given a connected QUDG G satisfying $\mathrm{R/r} \leq \sqrt{2}$, then Reactive-Virtual-Face-Traversal guarantees delivery between any source-destination pair $s, d \in V$.*

Proof. If $\|sd\| \leq \mathrm{r}$, then the packet is delivered directly. Otherwise, s and d reside in different cells. If d belongs to $\mathrm{Virt}(G_b)$, then the packet is guaranteed to be delivered. This holds due to the facts that Virtual-Face-Traversal guarantees message delivery on connected QUDGs satisfying $\mathrm{R/r} \leq \sqrt{2}$ [72, Corollary 5] and that G_b is a graph satisfying

these properties. Finally, if d does not belong to $\mathrm{Virt}(G_b)$, then there exists at least one other backbone node in its cell $C(d)$, which eventually receives and delivers the packet. In combination with the correctness of the algorithms BBB and BBN, this proves the claim. □

8.7 Discussion and Implications

This chapter introduces the first Ω-reactive local topology control algorithm (Reactive-Backbone) for construction of constant stretch topological planar spanners, as well as the first reactive guaranteed delivery geographic routing protocol (Reactive-Virtual-Face-Traversal) for restricted QUDGs. All previous local and localized techniques for network planarization and geographic routing on such graphs are beacon-based.

ReactiveBackbone is the first algorithm that is explicitly designed to construct a planar spanner only locally in the proximity of its executing node, whereas all previous approaches are either designed to be executed by all network nodes, or require additional assumptions on the minimum node distances.

This algorithm can be further extended to reactively construct a routing graph G'_r which is both a constant stretch topological spanner and a constant stretch Euclidean spanner, using the idea of shifted grids by Lillis et al. (see explanation in Section 8.1.2) combined with algorithm rPDT from Section 6.2. A node requiring its adjacency in G'_r computes the blue, red, and green grid cells to which it belongs. Then it starts three rounds of execution of algorithm rPDT and includes in the CTS the addressed grid cell. Only those nodes belonging to this grid cell participate in the reactive construction of PDT. After these three executions, the node is aware of its adjacent edges in three planar Euclidean spanners for the clique induced by the respective grid cells. Note that using an algorithm designed for unit disk graphs is feasible here, since the graph induced by the nodes of a grid cell is a unit disk graph for unit disk radius r. Lastly, the node proceeds with execution of algorithm ReactiveBackbone. After termination, it then knows its complete adjacency in G'_r. Although PDT is used for Euclidean spanner construction instead of BPS_2, as proposed by Lillis et al., the constant Euclidean spanning property of G'_r is maintained. The Euclidean spanning ratio of BPS_2 is smaller than 6.22, whereas the one of PDT is at most 7.98. Hence, this replacements affects the constant hidden in the $\mathcal{O}$-notation only marginally.

By showing ReactiveBackbone to be Ω-reactive, it immediately follows that it is at least as efficient regarding the volume of communication in terms of number of transmitted bits as any conventional, beacon-based algorithm that solves the problem at hand. In worst-case situations, their performances are equally poor. In all other scenarios the reactive solution actually helps to reduce the total number of message transmissions and hence, improves the resource efficiency of this topology control operation. At least theoretically, the gap in communication overhead between the reactive solution and any beacon-based counterpart can be made arbitrarily large, as shown by the example graph in Figure 8.4. Hence, conducting simulative experiments could merely help to quantify this gap.

Of course, it would be desirable to design an $\mathcal{O}$-reactive solution for the problem at hand, to further reduce the message overhead. For the case of the virtual planar backbone graph, which requires the computation of representative bridge edges, this is not possible as shown in Section 5.4. Hence, the present solution is "as reactive as possible" at least in the sense of the classification scheme of $\mathcal{O}$- and Ω-reactive algorithms.

The major drawback of ReactiveBackbone is best explained in comparison to algorithm rPDT (see Section 6.2). There, a node communicates at most with all its 1-hop neighbors to construct its desired local view. In contrast, in ReactiveBackbone even nodes that are 8 hops apart from the executing node may be involved in local view construction by sending messages. In other words, there is a considerable gap in the level of locality of these algorithms. Of course, this relates to the transition from unit disk to quasi unit disk model assumptions on the one hand, and to the chosen topology mapping on the other hand. Future research shall attempt to develop local topology mappings that reduce this gap, if possible at all.

Another problematic aspect of this algorithm concerns the choice of cluster heads and bridge edges; e.g., the node closest to its cell's geometric center becomes the cluster head. This choice is not unique as there could be two or more nodes at the same distance. In consequence, two messages would be sent at the very same time, which causes a collision. Similar situations can occur during the choice of bridge edges. Of course, this problem can be fixed by additionally considering a lexicographical ordering of node positions, or unique node ID's for breaking such ties. It is, however, non-trivial to integrate such a distinction in the distance based-delay functions used in the reactive constructions. In fact this is not a problem specific to this algorithm, but affects any algorithm making use of distance-based delay functions. Therefore, this problem is not further discussed here, but is addressed in a more general discussion on message collisions in reactive algorithms in Section 10.2 as part of the conclusion.

The beaconless computation of representative bridge edges, which is used as a subroutine in algorithm ReactiveBackbone, is of further use, independent of the algorithm presented here. All of the algorithms presented in [51,136,137,150,218–220] make use of rectangular and hexagonal geographical clusterings. Instead of computing the cell's adjacencies there based on 2-hop neighborhood information, the beaconless scheme can be used to reduce message overhead in these approaches.

Lastly, algorithm Reactive-Virtual-Face-Traversal is the first beaconless geographic routing protocol for constrained QUDGs that guarantees message delivery. The difference between this protocol and the underlying protocol Virtual-Face-Traversal by Guan [72, 73, 221] is that messages are virtually routed along edges of the virtual planar backbone $\mathrm{Virt}(G_b)$ instead of the virtual planar input graph. In the input graph, an edge can be intersected $\mathcal{O}(n^2)$ many times in the worst-case, i.e., an edge may contain $\mathcal{O}(n^2)$ virtual nodes. In contrast, in the reactive version of this protocol described here, messages are virtually routed along backbone edges and these are guaranteed to be intersected at most $\mathcal{O}(1)$ times. Therefore, the virtual routing path potentially consists of less routing steps to virtual nodes and thereby avoids detours, which leads to a more efficient operation.

Chapter 9

Local Planarization of Quasi Unit Disk Graphs in Asynchronous Systems

In previous chapters, the power of the reactive approach has been demonstrated. However, in order for these algorithms to be correct, simplifying assumptions are made.

In these algorithms it is assumed that message transmissions can be performed instantaneously in the following sense. Message m transmitted by a node u is received and processed by all neighbors of u simultaneously, such that the neighbors' delay timers are started synchronously, i.e., at the very same point in time. In fact, already very small differences in timing can lead to errors regarding correctness of the output. For instance, in algorithm rPDT, if not the closest but the second closest node responds first to a request due to non-synchronous start of delay timers, the requesting node's local view on the planar graph may be incorrect. This in turn possibly leads to incorrect routing decisions; in consequence, a message could be dropped or looped instead of being delivered correctly.

Furthermore, it is assumed that no retransmissions are required to transmit a message, i.e., it is assumed that a message m transmitted by node u is always correctly received by all neighbors of u on the first attempt. Without this assumption it could be the case that a neighbor does not (correctly) receive a message and in turn does not participate in the contention period, which would lead to the errors described above.

One application domain of reactive algorithms are wireless sensor networks. Such networks are inherently *asynchronous distributed systems*[1]. Firstly, hardware clocks are imperfect due to clock drift and therefore the nodes' clocks are not globally synchronized. In order to achieve the latter, an additional clock synchronization protocol (such as NTP, the well known *Network Time Protocol*) has to be executed, although at the cost of additional control and communication overhead. Secondly, due to link failures and collisions in the broadcast domain, link delays are generally unbounded. Hence, reactive algorithms which are relying on the aforementioned assumptions are likely to fail in such networks without additional synchronization.

Indeed, every synchronous distributed algorithm can be executed in every asynchronous environment by means of so called *synchronizers*. Synchronizers, however, impose addi-

[1] Two system models are generally distinguished in distributed computing: in *synchronous* systems, it is assumed that link delays are bounded and nodes have access to a global system clock; in *asynchronous* systems, link delays are assumed to be finite but unbounded and no global clock access is provided to the nodes (cf. [87]). A distributed algorithm is then said to be *(a)synchronous* if it is designed to work in an (a)synchronous system.

tional costs regarding time and message complexity, which depend on the network size (see e.g., [87, pp.69] for a detailed discussion). Hence, in the context of wireless sensor networks it is rather natural to solve algorithmic problems directly in an asynchronous manner.

This chapter is concerned with asynchronous solutions for (local) topology control on QUDGs. In contrast to previous chapters, the algorithm designed here is not reactive and—without additional assumptions on the minimum node distances—it is not even local. On the other hand, it neither requires the assumptions that messages are transmitted and processed instantaneously, nor that transmissions are reliable. It only assumes that sending of a message between two nodes succeeds eventually within finite time, e.g., by means of multiple retransmissions. Therefore, this chapter can rather be considered a digression and outlook on future research in reactive local topology control.

As pointed out in Section 3.2, QUDGs in which the minimum and maximum transmission radii differ cannot be planarized by edge removals without possibly disconnecting the graph. Due to this, there is a considerable body of work on distributed algorithms for QUDGs for computation of (nearly) planar backbone graphs and planar overlay graphs as surveyed in Section 3.2. Few of those algorithms are explicitly designed to operate in synchronous networks [70, 71, 139, 155]. Several papers [72, 84, 86, 149–151, 154, 156] leave this aspect unspecified. It is not clear if their solutions work under such conditions; deeper investigations and possibly modifications may be required. Only the approaches by Barrière et al. [83, 145] and Moaveninejad et al. [147, 148] are explicitly designed for use in asynchronous network environments. However, the planar routing graphs constructed by these algorithms suffer from large stretch factors. One reason for this is that they make use of GG for planarization of their overlay graphs and GG is known to impose a non-constant Euclidean spanning ratio, at least if used for unit disk graph planarization.

In this chapter, it is shown that the principle idea of these algorithms can be extended and improved such that the Euclidean stretch factor can be significantly decreased. The fully asynchronous distributed algorithm *AsyncPDT* introduced here constructs a connected and planar overlay graph for a given QUDG satisfying $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$. Hence, it enables local geographic routing with guaranteed delivery. The algorithm uses messages of size at most $\mathcal{O}(\mathrm{P}_{\max})$ bits, where $\mathrm{P}_{\max}$ denotes the number of bits required to represent a node's geographic position. Moreover, the algorithm is shown to be k-local if the input graph belongs to the class of λ-civilized graphs, i.e., if any two nodes have a Euclidean distance of at least $\lambda > 0$. Number k is a constant that only depends on λ and the transmission radii R and r of the network nodes.

In AsyncPDT fewer edges are removed since PDT is used for planarization rather than GG. Therefore, the resulting overlay graph is denser than that produced by the algorithm from Barrière et al.

This chapter is structured as follows. Section 9.1 introduces the network model, provides relevant definitions, and explains the approach by Barrière et al. [145] in more detail. The presentation of algorithm *AsyncPDT* follows in Section 9.2. Its correctness proof and analysis are provided in Section 9.3. Finally, in Section 9.4 advantages and disadvantages are discussed, and future research ideas are given.

9.1 Model, Preliminaries, and Fundamentals

Network Graph

Network graphs are modeled as quasi unit disk graphs $G = (V, E)$, over finite and distinct node sets $V \subset \mathbb{R}^2$.

Throughout this chapter it is assumed that G is a connected graph and that the ratio of the maximum to the minimum transmission radius is at most $\sqrt{2}$. Recall that this assumption guarantees that if two network edges uv and $u'v'$ intersect, then there exists an edge between one endpoint of uv and one endpoint of $u'v'$ (see Lemma 3.12).

As in previous chapters dealing with Delaunay triangulations, it is assumed that V is not collinear and not cocircular (cf. Assumptions 2.2 & 2.1).

System Model

Network nodes represent processing units that communicate via wireless multihop broadcasts. Nodes do not have access to a global system clock and possibly wake up asynchronously. Message transmissions are assumed to take arbitrary but finite time; i.e., messages are assumed to be transmitted eventually, possibly by means of multiple retransmissions. This model is known as the $\mathcal{ASYNC}$ model of distributed computing (cf. [87]) and it exists in both settings, with and without restrictions on message size. The approach presented here guarantees that message size is bounded in the order of number of bits required to represent a node's geographic position.

Asynchronous Computation of Gabriel Graphs

Given a QUDG $G = (V, E)$ satisfying $\mathrm{R/r} \leq \sqrt{2}$, from the viewpoint of some node $u \in V$, the algorithm from Barrière et al. [83, 145] proceeds as follows.

Initially, node u gathers full one-hop neighborhood information and stores it in its adjacency list $\mathcal{L}(u)$. Each neighbor that is added to this list is initially marked as *unprocessed*. Node u then enters the *Completion Phase* in which it processes its incident neighbors. It is assumed that every node is ready to receive messages and to update its adjacency list parallel to processing of its incident edges.

In the *Completion Phase*, node u processes each unprocessed neighbor $v \in \mathcal{L}(u)$ as follows. If there exists $w \in \mathcal{L}(u) \setminus \{u, v\}$ with $\mathrm{r} < \|vw\| \leq \mathrm{R}$ and $w \in \mathrm{D}(u, v)$, it ensures the creation of a *virtual edge* between v and w by sending them unicast messages containing the other node's position. When node u receives a message from (virtual) neighbor v containing position information about a node w with $uw \notin E$, then u adds w to $\mathcal{L}(u)$ with w marked as *unprocessed*, adds uw as a *virtual edge* and stores $\mathrm{ve}(u, w) = (u, v, w)$ as a path for *virtual routing* along this virtual edge. The result of this phase is a supergraph $S(G)$ of the input graph.

Sending of a message m via a (virtual) edge uw proceeds as follows. If w is a physical neighbor of u in G, then u simply forwards m to w. Otherwise, if w is a virtual neighbor of u, then u sends m according to $\mathrm{ve}(u, w) = (u, v, w)$. Note that the path contained by

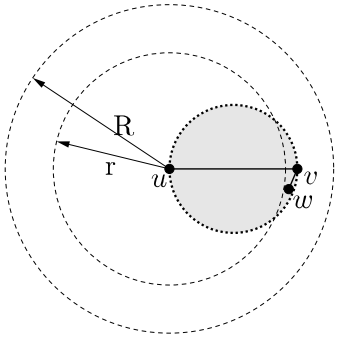


Figure 9.1: Example QUDG where the execution of the algorithm by Barrière et al. [83,145] yields an unconnected graph if full neighborhood information is not available during the first Completion Phase.

ve(u, w) may contain other virtual edges and possibly requires additional recursive steps for resolving the actual (physical) routing path.

Once all nodes in $\mathcal{L}(u)$ have been processed, node u starts the *Extraction Phase* in which node u discards all adjacent (virtual) edges $uv \in S(G)$ that violate the Gabriel criterion, i.e., if there exists a (virtual) neighbor w of u s.t. $w \in \mathrm{D}(u, v)$. The result is the graph $\mathrm{GG}(S(G))$.

After finishing the Extraction Phase, node u enters the *Routing Phase*. In case node u has some data packet to route, it uses any local guaranteed delivery geographic routing protocol, such as *GFG* [36], for forwarding this packet using the planar graph $\mathrm{GG}(S(G))$. Forwarding of the packet along a virtual edge is done as described above.

Messages about new virtual neighbors arrive asynchronously at their destinations. In particular it may happen that a node receives messages about new virtual neighbors although it has already entered the Extraction or Routing Phase, in which case this node returns to the Completion Phase and passes all phases again. However, the Completion and Extraction Phase terminate eventually. The result is then a connected, symmetric, and planar graph $\mathrm{GG}(S(G))$.

For correctness, the algorithm in its original description [83, 145] as well as in the summary given above, requires that a node's neighborhood in the input graph is known prior to the first execution of the Completion Phase by this node. To see this, consider the simple QUDG given in Figure 9.1. Suppose node v processes its incident edge vu without knowing its physical neighbor w. Then it does not send any message and virtual edge uw is never created. During the Extraction Phase the graph will lose its connectivity since edge vu is deleted by v.

This observation is problematic if the algorithm is executed in a strictly asynchronous environment, where the delivery of a message takes finite but arbitrary time. A node cannot know the point in time at which it has received all beacons from all neighboring nodes and consequently, it is not clear to that node when to start the Completion Phase. The algorithm introduced next avoids this problem and can be executed on completely asynchronous systems.

9.2 Asynchronous Computation of the Partial Delaunay Triangulation

Algorithm *AsyncPDT* is a modification of the algorithm from Barrière et al. [83, 145]. Essentially, instead of computing the Gabriel graph over $S(G)$, it computes the partial Delaunay triangulation over $S(G)$, denoted $\mathrm{PDT}(S(G))$. $\mathrm{PDT}(S(G))$ is a connected and planar supergraph of $\mathrm{GG}(S(G))$, whose Euclidean spanning ratio is significantly smaller compared to that of $\mathrm{GG}(S(G))$—at least on average in random scenarios.

The modifications concern the initialization as well as the Completion and Extraction Phase. The virtual routing process and the Routing Phase itself are equivalent.

Description of Algorithm AsyncPDT

In the following let u denote the node executing algorithm AsyncPDT. As in the original algorithm described above, it is assumed that parallel to the processing of incident edges, every node is ready to receive messages and to update its adjacency lists. A pseudocode representation of AsyncPDT is given in Algorithm 6.

Initialization Node u locally broadcasts $[\mathrm{beacon}, u]$ and initializes its *adjacency list* $\mathcal{L}(u)$ and *list of witnesses* $\mathcal{W}(u)$.

The adjacency list $\mathcal{L}(u)$ represents the node's adjacency in the supergraph $S(G)$. During further execution, $\mathcal{L}(u)$ will be complemented by new physical and virtual neighbors.

The witness list $\mathcal{W}(u)$ contains information about nodes at distance at most R from u, called *witnesses*, with which u does not share a (virtual) edge, but about which it has been informed by its (virtual) neighbors in order to guarantee symmetric removal decisions of nodes during the Extraction Phase. Witnesses can become virtual neighbors throughout the execution, but not vice versa. The algorithm ensures that the sets $\mathcal{L}(u)$ and $\mathcal{W}(u)$ are always disjoint.

Nodes in the lists $\mathcal{L}(u)$ and $\mathcal{W}(u)$ have a flag which indicates if they have already been *processed*. Whenever a node is added to any of these lists it is marked as *unprocessed* and is processed during the next execution of the *Completion Phase*.

Completion Phase While there exists an unprocessed (virtual) neighbor $v \in \mathcal{L}(u)$, node u processes it as follows.

For each (virtual) neighbor w with $\mathrm{r} < \|vw\| \leq \mathrm{R}$, such that either

(i) $w \in \mathrm{D}(u, v)$, or

(ii) $v \in \mathrm{D}(u, w)$ and w is marked as processed,

node u sends messages $[\mathrm{new}, w]$ to v and $[\mathrm{new}, v]$ to w. This guarantees that (virtual) edge vw is bidirectional. Moreover, part (ii) of this condition ensures creation of virtual edges even if not all physical neighbors are known prior to the first execution of the Completion Phase. This way the problem described in the context of the original algorithm (see Figure 9.1) is solved.

Algorithm 6 AsyncPDT
Variables: r and R refer to the minimum and maximum transmission radius, respectively.

Action by node u

1: Locally broadcast message $[\text{beacon}, u]$ ▷ Send beacon message
2: $\mathcal{L}(u) \leftarrow \{u\}$ with u marked as *processed* ▷ Initialize adjacency list
3: $\mathcal{W}(u) \leftarrow \emptyset$ ▷ Initialize list of witnesses
4: **Phase** Completion
5: **while** there exists an *unprocessed* node $v \in \mathcal{L}(u)$ **do**
6: **for** each $w \in \mathcal{L}(u) \setminus \{u\}$ **do**
7: **if** $\mathrm{r} < \|vw\| \leq \mathrm{R}$ **then**
8: **if** $w \in \mathrm{D}(u, v)$ or $v \in \mathrm{D}(u, w)$ and w is marked as *processed* **then**
9: Send message $[\text{new}, w]$ to v
10: Send message $[\text{new}, v]$ to w
11: **else**
12: Send message $[\text{info}, w]$ to v
13: Send message $[\text{info}, v]$ to w
14: **for** each $w \in \mathcal{W}(u)$ **do**
15: **if** $\mathrm{r} < \|vw\| \leq \mathrm{R}$ **then**
16: Send message $[\text{info}, w]$ to v
17: Mark v as *processed*
18: **while** there exists an *unprocessed* node $\hat{v} \in \mathcal{W}(u)$ **do**
19: **for** each $w \in \mathcal{L}(u) \setminus \{u\}$ **do**
20: **if** $\mathrm{r} < \|\hat{v}w\| \leq \mathrm{R}$ **then**
21: Send message $[\text{info}, \hat{v}]$ to w
22: Mark $\hat{v}$ as *processed*
23: **Phase** Extraction
24: $\mathrm{E}^{\mathrm{PDT}} \leftarrow \emptyset$ ▷ Initialize output
25: **for** each (virtual) neighbor $v \in \mathcal{L}(u)$ **do**
26: Select the angle maximizing node $w \in \mathcal{L}(u)$ w.r.t. uv
27: **if** $w \notin \mathrm{D}(u, v)$ **then**
28: $\mathrm{E}^{\mathrm{PDT}} \leftarrow \mathrm{E}^{\mathrm{PDT}} \cup \{uv\}$ ▷ uv belongs to GG
29: **else**
30: **if** $\mathrm{C}(u, v, w) \cap (\mathcal{L}(u) \cup \mathcal{W}(u)) = \{u, v, w\} \wedge \sin(\measuredangle uwv) \geq \|uv\|/\mathrm{R}$ **then**
31: $\mathrm{E}^{\mathrm{PDT}} \leftarrow \mathrm{E}^{\mathrm{PDT}} \cup \{uv\}$ ▷ uv belongs to PDT $\setminus$ GG

Message handling by node u

32: **on** reception of $[\text{beacon}, v]$ from v **do**
33: **if** $v \notin \mathcal{L}(u)$ **then**
34: $\mathcal{L}(u) \leftarrow \mathcal{L}(u) \cup \{v\}$ and mark v as *unprocessed*
35: **else** ▷ Node v is a physical neighbor, not a virtual one
36: Remove $\mathrm{ve}(u, v)$
37: $\mathcal{W}(u) \leftarrow \mathcal{W}(u) \setminus \{v\}$ ▷ In case v is currently known as a witness
38: **on** reception of $[\text{new}, w]$ from v **do**
39: **if** $w \notin \mathcal{L}(u)$ **then**
40: $\mathcal{L}(u) \leftarrow \mathcal{L}(u) \cup \{w\}$ and mark w as *unprocessed*
41: $\mathrm{ve}(u, w) \leftarrow (u, v, w)$
42: $\mathcal{W}(u) \leftarrow \mathcal{W}(u) \setminus \{w\}$ ▷ In case v is currently known as a witness
43: **on** reception of $[\text{info}, w]$ from v **do**
44: **if** $w \notin \mathcal{L}(u) \cup \mathcal{W}(u)$ **then**
45: $\mathcal{W}(u) \leftarrow \mathcal{W}(u) \cup \{w\}$ and mark w as *unprocessed*

Furthermore, for each (virtual) neighbor w with $\mathrm{r} < \|vw\| \leq \mathrm{R}$ and $w \notin \mathrm{D}(u, v)$, node u sends messages $[\mathrm{info}, w]$ to v and $[\mathrm{info}, v]$ to w, which ensures symmetric edge removal decisions during the Extraction Phase.

In addition, in order to facilitate symmetric edge removal decisions, for every witness node $w \in \mathcal{W}(u)$ with $\mathrm{r} < \|vw\| \leq \mathrm{R}$, node u sends message $[\mathrm{info}, w]$ to v. Note that w is not a (virtual) neighbor of u and it is therefore not being sent a message.

Once all unprocessed (virtual) neighbors have been processed, all unprocessed witnesses $\hat{v} \in \mathcal{W}(u)$ are processed as follows. For each (virtual) neighbor w with $\mathrm{r} < \|\hat{v}w\| \leq \mathrm{R}$ node u simply sends message $[\mathrm{info}, \hat{v}]$ to w. This additional while-loop accounts for the scenario that the (virtual) neighbor w of u has already been processed when u learns about a witness $\hat{v}$ with $\mathrm{r} < \|\hat{v}w\| \leq \mathrm{R}$.

Message handling Node u handles incoming messages as follows.

On reception of $[\mathrm{beacon}, v]$ from v, node u adds v to its adjacency list $\mathcal{L}(u)$ and marks it as unprocessed, unless v is already contained in that list. In this case v is currently known as a virtual neighbor and the corresponding virtual routing path can be removed. Moreover, in the case that v is currently known as a witness, it is removed from $\mathcal{W}(u)$.

On reception of $[\mathrm{new}, w]$ from v, node u checks if w is already contained in $\mathcal{L}(u)$. If this is not the case, u adds w to $\mathcal{L}(u)$, marks it as unprocessed, and stores the virtual routing path $\mathrm{ve}(u, w) = (u, v, w)$. If w is currently a witness node (i.e., $w \in \mathcal{W}(u)$), it is removed from this list.

On reception of $[\mathrm{info}, w]$ from neighbor v, node u checks if w is already contained in $\mathcal{L}(u) \cup \mathcal{W}(u)$. If not, w is added to the list of witnesses $\mathcal{W}(u)$ and is marked as unprocessed.

Extraction Phase The set of PDT edges adjacent to u, denoted $\mathrm{E}^{\mathrm{PDT}}$, is initialized to be the empty set. Then, for each (virtual) neighbor v, node u checks if uv is contained in $\mathrm{PDT}(S(G))$, based on the current state of the sets $\mathcal{L}(u)$ and $\mathcal{W}(u)$. If so, it simply adds it to $\mathrm{E}^{\mathrm{PDT}}$. Note that set $\mathrm{E}^{\mathrm{PDT}}$ is computed from scratch each time a node (re)enters this phase. This ensures that former (possibly wrong) decisions that were based on outdated information are not taken over. After termination of this phase, node u may enter the Routing Phase. However, as soon as it learns about new (virtual) neighbors or witnesses, it returns to the Completion Phase and the Extraction Phase thereafter.

9.3 Correctness and Analysis

Termination and Symmetry

The proofs of Lemma 9.1, 9.3, & 9.5 are almost equivalent to the proofs of Lemma 1, 2, & 4 in [145] and are given here for completeness. The proof of Lemma 9.4 is partially similar to the proof of Lemma 3 in [145].

Lemma 9.1. *If a node u learns about a virtual neighbor v, then v eventually learns about the existence of u. Furthermore, the virtual routing protocol ensures the correct delivery of messages through virtual edges.*

Proof. Consider any virtual edge $uv \in S(G)$. Node u learns about its virtual neighbor v during processing of edge e by some node $w \neq u, v$, where either $e = uw$ and $v \in \mathrm{D}(u, w)$, or $e = vw$ and $u \in \mathrm{D}(v, w)$. These cases are symmetric and hence, it can be assumed w.l.o.g. that $e = uw$. Then, wv is an edge in G. If $v \in \mathcal{L}(w)$ while w processes u, then w sends new-neighbor messages to u and v, which creates the virtual edge uv. Otherwise, u has already been processed once w processes v. Because $v \in \mathrm{D}(u, w)$ and u is marked as processed, w sends new-neighbor messages to u and v, which creates the virtual edge uv.

Showing that virtual routing protocol ensures the correct delivery of messages through virtual edges is identical to the corresponding part of the proof of Lemma 1 in [145].

By construction, for any (virtual) edge $uv \in S(G)$ there are two paths associated with it. If $uv \in E$, then $\Pi(u, v) = \langle u, v \rangle$ and $\Pi(v, u) = \langle v, u \rangle$. If uv is a virtual edge, then

- $\Pi(u, v) = \langle \Pi(u, w) \oplus \Pi(w, v) \rangle$, where w is the node from which u learnt about v, and $\oplus$ is the concatenation of two paths; and
- $\Pi(v, u) = \langle \Pi(v, w') \oplus \Pi(w', u) \rangle$, where w' is the node from which v learnt about u.

The paths $\Pi(u, v)$ and $\Pi(v, u)$ may not be symmetric due to asynchronous message arrivals. Finally, it is shown that the virtual routing protocol as described above routes a message m from u to v along $\Pi(u, v)$. The proof is given by induction the length l of the path $\Pi(u, v)$.

I.h.: The message is successfully transmitted along a path of length l, for an arbitrary but fixed length l.

B.c. ($l = 1$): Edge uv is an edge from the input graph and the message is sent along this link.

I.s. ($l \to l + 1$): Edge uv is a virtual edge and node u sends m along $\mathrm{vw}(u, v) = (u, w, v)$. By the i.h., m eventually arrives at w, where it is forwarded from w to v. Again by the i.h. this message arrives eventually at v. □

Lemma 9.2. *If applied onto the same input graph, the supergraphs produced by the algorithms by Barrière et al. [83, 145] and by AsyncPDT are equivalent.*

Proof. In the following let A and B denote the supergraphs constructed by the algorithms by AsyncPDT and Barrière et al. [83, 145] over the same input graph $G = (V, E)$. Note that A and B can only differ w.r.t. their virtual edges.

Let $uv \in B$ be any virtual edge. It has been created during processing of an edge e by some node $w \neq u, v$. Using the same line of argumentation as in the first part of the proof of Lemma 9.1, it follows that $uv \in A$ holds.

Now, let $uv \in A$ be any virtual edge. It has been created during processing of one of the edges e and e', where either $e = uw$, $e' = vw$, and $v \in \mathrm{D}(u, w)$, or $e = vw$, $e' = uw$, and $u \in \mathrm{D}(v, w)$. In both cases it holds that e' is an edge from the input graph. According to the algorithm by Barrière et al., all physical edges incident to a node are known to it

once it enters the Completion Phase for the first time. Therefore, during processing of edge e the virtual edge uv will be constructed. □

Lemma 9.3. *The Completion Phase terminates, that is, all nodes eventually complete the Completion Phase and never come back to it.*

Proof. Each node u processes at most $n_u = V \cap \mathrm{C_R}(u) - 1$ many other nodes. The processing of a node causes sending of a finite number of messages to a finite number of nodes and these messages arrive eventually by Lemma 9.1. Furthermore, each such node is processed exactly once. □

Lemma 9.4. *The graph obtained in the Extraction Phase is symmetric, that is, if a node u decides to remove the (virtual) edge uv, then v eventually removes it as well.*

Proof. Let $uv \in S(G)$ be any (virtual) edge and assume u removes it during the Extraction Phase. Then there is at least one node in $\mathrm{D}(u, v)$. Let w be the angle maximizing node w.r.t. uv.

If $\|vw\| \leq \mathrm{r}$, then v receives the beacon from w eventually. Otherwise, if $\|vw\| > \mathrm{r}$ two cases have to be considered. Since u is currently in the Extraction Phase, it has already processed edge uv in the Completion Phase. At this point in time either $w \in \mathcal{L}(u)$, or not. In the former case, u has send new-neighbor messages to v and w and by Lemma 9.1 these arrive eventually. In the latter case, at some point in time after processing edge uv, node u learns about node w and processes it. During this process, node u sends the new-neighbor messages to v and w, and these messages arrive eventually. Hence, in any case, node v learns about node w.

If $\sin(\measuredangle uwv) < \|uv\|/\mathrm{R}$ holds, then v eventually removes edge uv. Otherwise, if $\sin(\measuredangle uwv) \geq \|uv\|/\mathrm{R}$ holds, then there must exists $\hat{w} \in \mathcal{L}(u) \cup \mathcal{W}(u)$ with $\hat{w} \in \mathrm{C}(u, v, w)$ and $\hat{w} \neq u, v, w$. Note that under these assumptions $\|\hat{w}v\| \leq \mathrm{R}$ holds.

Assume $\|\hat{w}v\| > \mathrm{r}$ holds (otherwise, v receives the beacon from $\hat{w}$ anyway and removes uv) and consider the point in time when u processes $\hat{w}$. By that time, either $v \in \mathcal{L}(u)$ or $v \notin \mathcal{L}(u)$. If the former holds, then u sends the coordinates of $\hat{w}$ to v. Otherwise, $\hat{w}$ has already been processed, when u processes v. In the course of this, u sends the coordinates of $\hat{w}$ to v.

Either way, by Lemma 9.1 these messages arrive eventually and v decides to remove uv during the Extraction Phase. □

Lemma 9.5 (Lemma 4 in [145])**.** *The Extraction Phase terminates, that is, all nodes eventually complete the Extraction Phase, and never come back to it.*

Proof. During the Extraction Phase, a node processes each of its incident (virtual) neighbors, which is a finite number, exactly once based on its local knowledge. Although the Extraction Phase may be executed multiple times, by Lemma 9.3, the Completion Phase terminates eventually. Then, the Extraction Phase is executed one more time. After this, the algorithm never comes back to it. □

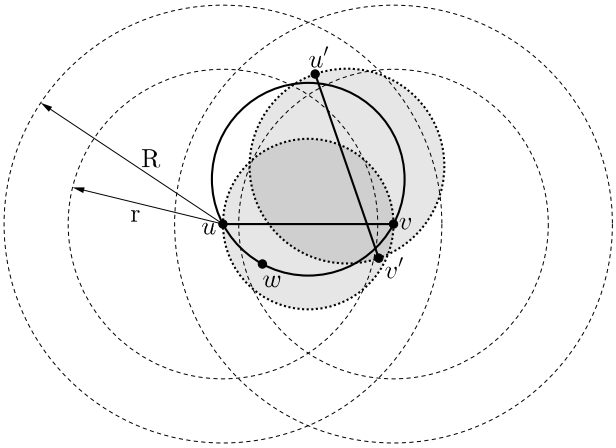


Figure 9.2: Illustration for the proof of Theorem 9.8. Shaded circles represent Gabriel circles, dashed circles represent transmission radii r and R, and the bold printed black circle represents $\mathrm{C}(u, v, w)$.

Connectivity, Planarity, and Euclidean Spanning Ratio

Theorem 9.6 (Subgraph relation)**.** *For any QUDG* $G = (V, E)$ *with* $\mathrm{R/r} \leq \sqrt{2}$*, it holds that* $\mathrm{PDT(S}(G))$ *is a supergraph of* $\mathrm{GG}(S(G))$*, i.e.,* $\mathrm{GG}(S(G)) \subseteq \mathrm{PDT(S}(G))$.

Proof. Let $uv \in \mathrm{GG}(S(G))$ be any edge. Then $\mathrm{D}(u, v)$ does not contain any (virtual) neighbor of u or v. In this case, edge uv will be added to the output sets of u and v during the Extraction Phase of algorithm AsyncPDT. □

Theorem 9.7 (Connectivity)**.** *If* $G = (V, E)$ *is a connected* QUDG *with* $\mathrm{R/r} \leq \sqrt{2}$*, then* $\mathrm{PDT(S}(G))$ *is a connected graph.*

Proof. According to Lemma 5 in [145], $\mathrm{GG(S}(G))$ is connected, if G is connected and $\mathrm{R/r} \leq \sqrt{2}$ holds. Since $\mathrm{GG(S}(G)) \subseteq \mathrm{PDT}(S(G))$ by Theorem 9.6, the claim follows. □

Theorem 9.8 (Planarity)**.** *If* $G = (V, E)$ *is a* QUDG *with* $\mathrm{R/r} \leq \sqrt{2}$*, then* $\mathrm{PDT}(S(G))$ *is a planar graph.*

Proof. The claim is proven by contradiction. For an illustration, see Figure 9.2.

Let $uv, u'v' \in \mathrm{PDT}(S(G))$ be two intersecting (virtual) edges. Consider the quadrangle formed by the nodes u, u', v, v'. At least one of the interior angles of this quadrangle is at least $\pi/2$. It can be assumed w.l.o.g. that $\measuredangle uv'v \geq \pi/2$. Then, v' is contained in $\mathrm{D}(u, v)$ and hence, $uv \in \mathrm{PDT}(S(G)) \setminus \mathrm{GG}(S(G))$. This implies that the half circle of $\mathrm{D}(u, v)$, which is bounded by $\overline{uv}$ and does not contain v', is empty of other nodes from V. Moreover, there exists a circle $\mathrm{C}(u, v, w)$, where w is the angle maximizing node w.r.t. uv, s.t. $\sin(\measuredangle uwv) \geq \|uv\|/\mathrm{R}$ and this circle is empty of (virtual) neighbors and witnesses known to u and v. Note that v' and w are located in the same half-circle of $\mathrm{D}(u, v)$ w.r.t. uv.

With $v' \in \mathrm{D}(u,v)$ and the assumption that the input graph satisfies $\mathrm{R/r} \leq \sqrt{2}$, it holds in addition that $v' \in \mathrm{N}_1^G(u)$ or $v' \in \mathrm{N}_1^G(v)$. It can be assumed w.l.o.g. that $v' \in \mathrm{N}_1^G(v)$. Furthermore, because uv and $u'v'$ intersect, and V is assumed to be non-collinear (i.e., neither v' nor u' are located on uv), u' must be located on the other side of $\ell(u,v)$ w.r.t. v'. Distinguish the following two cases.

Case ($u' \in \mathrm{C}(u,v,w)$): With $\sin(\measuredangle uwv) \geq \|uv\|/\mathrm{R}$ it holds that the diameter of $\mathrm{C}(u,v,w)$ is at most R. Hence, $\|uu'\|, \|vu'\| \leq \mathrm{R}$ holds. But then, by the time node v' processed edge $v'u'$, node v' would have informed its neighbor v about the position of u'. Hence, u' would be a witness or a (virtual) neighbor of v and uv would have been removed during the Extraction Phase, which contradicts the assumption that $uv \in \mathrm{PDT}(S(G))$.

Case ($u' \notin \mathrm{C}(u,v,w)$): Since u' is outside $\mathrm{C}(u,v,w)$ and on the opposite site of $\ell(u,v)$ w.r.t. w, the application of Thales' theorem yields $\measuredangle uu'v + \measuredangle uwv < \pi$. As w is assumed to be the angle maximizing node w.r.t. uv, it holds that $\measuredangle uv'v \leq \measuredangle uwv$ and hence, $\measuredangle uu'v + \measuredangle uv'v < \pi$. But then, one of the two interior angles $\measuredangle u'uv'$ and $\measuredangle u'vv'$ of the quadrangle formed by u, u', v, v' must be at least $\pi/2$. Hence, $u \in \mathrm{D}(u',v')$ or $v \in \mathrm{D}(u',v')$ must hold.

Assume w.l.o.g. that $v \in \mathrm{D}(u,v')$. In particular this implies $u'v' \in \mathrm{PDT}(S(G)) \setminus \mathrm{GG}(S(G))$. Since u' is strictly outside $\mathrm{C}(u,v,w)$, v' is on the boundary of (in case $v' = w$) or outside $\mathrm{C}(u,v,w)$, and u and v are separated by $\ell(u',v')$, circle $\mathrm{C}(u',v',v)$ must contain u and w. But then, this holds in particular for circle $\mathrm{C}(u',v',w')$, where $w' \in V$ is the angle maximizing node w.r.t. $u'v'$. Since $v' \in \mathrm{D}(u,v)$, $\|uv'\| \leq \mathrm{R}$ holds and hence, v would have informed its neighbor v' about the position of u either while processing uv, or while processing vv'. But then, v' would have removed edge $u'v'$ during the Extraction Phase which contradicts the assumption that $u'v' \in \mathrm{PDT}(S(G))$. □

In the following, assign any physical edge uv a weight equal to its Euclidean length $\|uv\|$ and any virtual edge uv a weight equal to $\max\{||\mathrm{ve}(uv)||, ||\mathrm{ve}(vu)||\}$, i.e., the maximum Euclidean path length of the two paths when using the virtual routing protocol between u and v, and vice versa.

Theorem 9.9 (Euclidean spanning ratio)**.** *The Euclidean spanning ratio of* $\mathrm{PDT}(S(G))$ *is at most as large as the Euclidean spanning ratio of* $\mathrm{GG}(S(G))$.

Proof. The claim follows immediately from Theorem 9.6 proving that $\mathrm{GG}(S(G))$ is a subgraph of $\mathrm{PDT}(S(G))$. □

The author of this thesis presented in [78] (in a joint work with Frey) preliminary simulation results on the average Euclidean spanning ratios of $\mathrm{GG}(S(G))$ and $\mathrm{PDT}(S(G))$. It is shown that the average Euclidean spanning ratio of $\mathrm{PDT}(S(G))$ is roughly 1.5, independent of the network density (average node degree). On average, the improvement of the Euclidean spanning ratio of $\mathrm{PDT}(S(G))$ compared to $\mathrm{GG}(S(G))$ is 21 %.

Message Size

Under the assumption that no two nodes have the exact same position, a node can be uniquely identified by its position. Any message sent during algorithm execution contains

at most a constant number of control bits and three node positions: namely, the positions of the origin and destination, as well as the position of a single witness or virtual neighbor. Hence, any message sent during algorithm execution is of size $\mathcal{O}(\mathrm{P}_{\max})$ bits, where $\mathrm{P}_{\max}$ is the number of bits needed to represent a node's geographic position.

Discussion of Locality

Next, the locality of the proposed approach is discussed. Recall that the supergraph constructed by executing the Completion Phase is identical to the supergraph obtained by the Completion Phase of the algorithm by Barrière et al. [145].

In Proposition 1 in [145] it is shown that if any two nodes in the input QUDG G with $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$ have a Euclidean distance of at least λ, then the hop-distance of the route in G corresponding to a virtual edge in $S(G)$ is at most $1 + (\mathrm{R}^2 - \mathrm{r}^2)/\lambda^2$. With $\mathrm{R} = 1$ and $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$ it follows that any virtual edge has a hop-length of at most $1 + 1/(2\lambda^2)$ (see also Lemma 8.3 in [86]).

Let uv be a virtual edge in $S(G)$ and $k = \lceil 1 + 1/(2\lambda^2) \rceil$. In order to construct this virtual edge uv, both u and v require at most k-hop neighborhood information in G. However, as shown next, it may take $c \cdot k$-hop neighborhood information, for a constant $c < 9$, in order for u and v to decide if edge uv belongs to $\mathrm{PDT}(S(G))$.

Lemma 9.10. *If in G any two nodes have minimum Euclidean distance $\lambda > 0$, for any node u, the decision of whether any of its adjacent (virtual) edges uv in $S(G)$ is contained in $\mathrm{PDT}(S(G))$ depends on at most the $c \cdot k$-neighborhood of u for some constant $c < 9$ and $k = \lceil 1 + 1/(2\lambda^2) \rceil$.*

Proof. Consider an arbitrary edge $uv \in S(G)$ which is not contained in $\mathrm{PDT}(S(G))$. Then there exists at least one node from $V \setminus \{u, v\}$ in $\mathrm{D}(u, v)$. Let $w \in \mathrm{D}(u, v)$ be the angle maximizing node w.r.t. uv. With $\mathrm{R}/\mathrm{r} \leq \sqrt{2}$ it holds that $w \in \mathrm{N}_1^G(u)$ or $w \in \mathrm{N}_1^G(v)$. Hence, it holds that $v, w \in \mathrm{N}_k^G(u)$.

If $\sin(\measuredangle uwv) < \|uv\|/\mathrm{R}$ holds, then u can immediately decide that $uv \notin \mathrm{PDT}(S(G))$ based on set $\mathrm{N}_k^G(u)$, which includes the positions of v and w.

For the remaining case, where $\sin(\measuredangle uwv) \geq \|uv\|/\mathrm{R}$, consider Figure 9.3.

Because $uv \notin \mathrm{PDT}(S(G))$, circle $\mathrm{C}(u, v, w)$ contains at least one other node in its interior, which lies on the opposite side of w w.r.t. $\ell(u, v)$. Let $\hat{w}$ be any such node and assume that u is not a (virtual) neighbor of $\hat{w}$ (otherwise, u can decide that $uv \notin \mathrm{PDT}(S(G))$ holds based on its k-hop neighborhood information).

There must exist a (virtual) neighbor of u that has sent an info-message about the position of $\hat{w}$ to u. Recall that for any node a, to send an info-message about the position of a node c to a node b, a and b must be (virtual) neighbors and the Euclidean distances $\|ac\|$ and $\|bc\|$ are at most R. This (virtual) neighbor itself may have received an info-message about the existence of $\hat{w}$, along a (virtual) edge, and so forth.

This means that the longest possible hop-path in G along which the position of node $\hat{w}$ may be passed on is a sequence of virtual neighbors $\langle u = u_1, v = u_2, ..., u_c = \hat{w} \rangle$ being

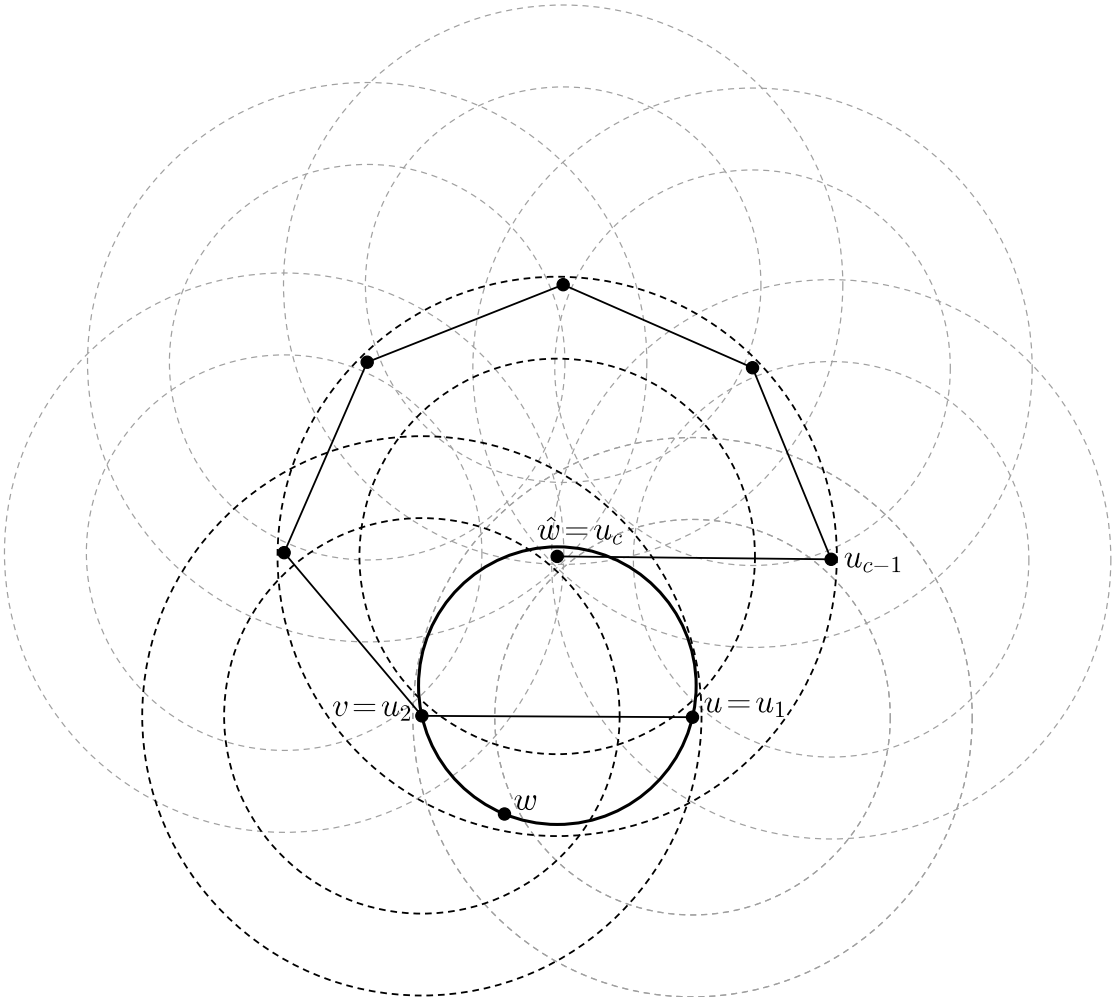


Figure 9.3: Illustration for the proof of Lemma 9.10. Dashed circles represent the transmission radii r and R, solid lines represent virtual edges, and the solid circle represents $\mathrm{C}(u, v, w)$.

contained in $\mathrm{C_R}(\hat{w})$. As the distance between any two such virtual neighbors u_i, u_{i+1} is strictly larger than r, this path consists of at most

$$c < \frac{2\pi \mathrm{R}}{\mathrm{r}} + 1 - 1 \tag{9.1}$$

$$\leq 2\pi\sqrt{2} \approx 8.89 \tag{9.2}$$

virtual edges. In Equation 9.1, the minus one accounts for the virtual edge missing between u_{c-1} and u_1 (otherwise, there would be a shorter path via the (virtual) edge $u_{c-1}u_1$), whereas the plus one accounts for the virtual edge $u_{c-1}u_c$. Hence, the information about the position of $\hat{w}$ is passed on along a path consisting of at most $k \cdot c$ hops in the input QUDG G. □

One direct implication of the above lemma is that for λ-civilized QUDGs satisfying $R/r \leq \sqrt{2}$, AsyncPDT is a *ck*-local view topology control algorithm as defined in Definition 5.3. Hence, for computation of a node's local view on PDT($S(G)$) algorithm AsyncPDT can be altered in the following manner while maintaining its correctness.

Some node u that requires its local view on a connected and planar subgraph, e.g., in case of a local minimum situation, node u starts executing AsyncPDT. The messages of type *new* and *info* it sends to its neighbors, contain additionally a *time-to-live* (TTL) equal to *ck*. Upon receiving such messages, the one-hop neighbors also start execution and send messages along with TTL $ck - 1$, and so on. Every node receiving a message containing a TTL larger than zero participates in the planar graph construction.

9.4 Discussion

The proposed approach has several advantages and disadvantages in particular in comparison to the other asynchronous approaches by Barrière et al. [83, 145] and Moaveninejad et al. [147, 148].

Like the aforementioned approaches, algorithm AsyncPDT is only local under the assumption that any two nodes are at least some Euclidean distance $\lambda > 0$ apart from each other. This minimum distance assumption appears to be reasonable considering the fact that nodes have a physical extent. Hence, the algorithm scales well with increasing network size.

Furthermore, compared to the previous approaches, AsyncPDT yields the densest output graphs and outperforms the approach by Barrière et al. regarding Euclidean spanning ratio. For certain applications, such as local geographic routing, this is actually advantageous. Shorter routing paths can contribute to more resource efficient routing operations, which is strongly desirable, in particular with respect to the resource constraints wireless sensor networks.

However, the advantage of short routing paths in PDT($S(G)$) comes at the cost of increased message complexity, which can be considered the major drawback of the proposed algorithm.

One reason for the high message complexity is that a node possibly sends multiple messages to one neighbor at a time. It is a straightforward extension of the proposed algorithm to encapsulate the information of these messages into a single message at the cost of increased message size. However, this solution only shifts the problem rather than tackles it. Nevertheless, there are two solutions for actual reduction of message overhead that are explained next.

Currently, during algorithm execution, it is possible that a node u sends k unicast messages to inform its neighbors $n_1, .., n_k$ about a witness node x. Instead it would suffice if u locally broadcasts the position of x only once. Based on their own, the sender's, and the witness' position, nodes $n_1, ..., n_k$ can easily compute whether or not they are addressed by this broadcast.

The other solution is based on ideas of Moaveninejad et al. [148]. Before entering the Completion Phase, they remove short non-Gabriel edges of length $\leq r$. This preliminary

edge removal yields a sparser graph with fewer edge intersections, which helps to reduce the message overhead produced during the Completion Phase. The Gabriel graph is known to have a worst-case Euclidean spanning ratio of $\Theta(\sqrt{n})$ [109] and removing these edges leads to a higher spanning ratio. Instead, short non-PDT edges should be removed, as PDT is a constant stretch Euclidean spanner of the unit disk graph and removal of such edges does not increase pairwise node distances significantly. In contrast to the aforementioned extensions, this algorithm modification invalidates some of the proofs. Additional work for guaranteeing its correctness is required.

Finally, as already pointed out by Barrière et al. [83, 145], all these algorithms have a problem with termination detection. Since messages are assumed to arrive asynchronously within finite but unpredictable time, for node u it is not clear at which point in time its Extraction Phase has finally terminated. In this model it may always be the case that some message is still in delivery. This problem is generally non-trivial.

One possible heuristic solution is the use of a time-based threshold parameter t_{th}. That is, node u assumes that it does not receive any more messages if it has not received a message within the last t_{th} time steps. This threshold parameter t_{th} depends on the level of asynchrony and could be chosen based on experiments in a pre-processing stage. However, this approach cannot guarantee correctness.

Chapter 10

Conclusion

Energy and bandwidth are severely constrained resources in wireless ad hoc and sensor networks [6]. Moreover, depending on the application scenario, such networks are large-scale and consist of thousands of nodes [9] or even more, as in the vision of "Smart Dust" [10]. Therefore, scalability and resource efficiency are key challenges in algorithm design for such networks.

Reactive local algorithms are particularly well suited to these demands. Firstly, they are local distributed algorithms which scale well with increasing network size since computations of nodes involve only those nodes that are at most a constant number of hops apart. Secondly, these algorithms are reactive (aka *beaconless* [16] or *contention-based* [17]). Reactive algorithms reduce resource consumption for a given task by consistently avoiding unnecessary message transmissions and communication overhead. In contrast to conventional, beacon-based, distributed algorithms, nodes do not set up or maintain complete neighborhood tables in reactive algorithms. Instead, only a small subset of a node's neighborhood actively participates in problem solving by using "blind" local broadcasts and timer-based contentions. Neighbors that are not eligible to solve the problem stay passive and do not transmit any message. This way, the overall communication load on the network is reduced (see, e.g., [23]), which has various positive effects: Interference and collision probability are reduced, which leads to an increase in packet reception ratio and a decrease in latencies. Available bandwidth for user data is saved. And finally, topology information gathered during algorithm execution is more current and less likely to be inaccurate, which is the main cause of packet loss in uncongested networks [12].

In this sense, reactive algorithms support the common position [7, 8] that *silence is golden* in wireless ad hoc and sensor networks.

Apart from theoretical worst-case scenarios, the aforementioned advantages have been verified empirically by means of simulations [18–22] as well as in actual testbed experiments [23–25].

In present research, the reactive approach is used to design resource efficient local algorithms for geographic routing and topology control, in particular the construction of connected and planar representations of given network graphs. Both geographic routing and network planarization are two important basic functionalities for wireless ad hoc and sensor networks. The former enables wireless multi-hop communication in the absence of any network infrastructure, based on geographic node positions. The latter provides sufficient conditions for efficient local solutions to a variety of algorithmic problems (see the list provided in Section 1.1).

The deficient state of research concerning reactive geographic routing and topology control algorithms led to the three primary objectives of this thesis, listed as Objectives 1–3 in Section 1.2. In Section 10.1 it is now discussed to what extent these objectives have been achieved in this thesis. Subsequently, in Section 10.2 the contributions are further discussed and new research directions are outlined. This thesis closes in Section 10.3 with some final remarks.

10.1 Comparison of Research Objectives and Results

Objective 1: Foundation of reactive local topology control Reactive topology control is a promising but immature research area. On the one hand, existing algorithms as well as those introduced in this work show that they can significantly improve message efficiency of topology control operations. On the other hand, the term *reactive topology control* lacks conceptual and mathematical clarity. Prior to this work, answering the following questions was tedious or even impossible: What formally distinguishes a reactive topology control algorithm from a conventional distributed one? What is a "good" or efficient reactive topology control algorithm? Which problems can or cannot be solved by such algorithms in general?

The concepts of $\mathcal{O}$- and Ω-reactive topology control introduced in Chapter 5 provide a means to answer all of these questions. In the first place, these definitions clearly distinguish the class of reactive from conventional, beacon-based, distributed algorithms in terms of worst-case message complexity. This is a reasonable choice, since message complexity is in general a key metric for quantifying the quality of distributed algorithms. Secondly, these algorithm classes facilitate a taxonomy of all known and future algorithms of this kind. This helps to sort contributions to this field and to expose research gaps. These definitions are also held general enough to capture not only planar graph constructions, but all aspects of topology control. Lastly, these definitions enable in-depth investigation of the principal power of the reactive approach, beyond pure algorithm design. This has been demonstrated by proving two impossibility results regarding the reactive computability of well-know topology control structures.

For these reasons, Objective 1 is essentially achieved. The concept itself is, however, a reasonable object for future investigation since in its current form it does not capture the stochastic nature of localization and wireless communication.

Objective 2: Reactive algorithms for construction of planar spanners There is an evident demand for reactive local topology control algorithms for construction of a node's adjacency in connected, planar constant stretch Euclidean and topological spanners. Such algorithms would contribute to more efficient solutions to the various algorithmic problems described in [35–56]. Prior to this work, there were only reactive local topology control algorithms for construction of planar graphs, whose Euclidean and topological spanning ratios are dependent on the number of nodes in the network [18, 107, 187, 188]. Moreover, these algorithms guarantee correctness only for networks obeying unit disk graph properties. A significant part of this thesis is dedicated to reactive local planar

Euclidean and topological spanner constructions under unit disk and the less restrictive quasi unit disk graph model assumptions.

Under unit disk graph model assumptions, in Chapter 6, it is shown that it is indeed possible to reactively construct Euclidean spanners. The $\mathcal{O}$-reactive local topology control algorithm rPDT introduced there constructs a node's local view on the partial Delaunay triangulation (PDT), which is proven to be a connected planar Euclidean spanner of the unit disk graph. This algorithm is not only the first reactive algorithm for construction of Euclidean spanners, but also the first reactive topology control algorithm that is message optimal. It completely avoids message transmissions by nodes that are not part of the problem's solution. In addition, the proof that PDT is a Euclidean spanner has important implications regarding Objective 3 as well as for its applications in literature. On the downside, PDT is not a constant stretch topological spanner for the unit disk graph, and is not necessarily connected under the more realistic quasi unit disk model assumptions.

The Ω-reactive local algorithm *ReactiveBackbone* introduced in Chapter 8 remedies the situation. It reactively computes a node's local view on a connected routing graph, which consists of a virtual planar backbone. The routing graph itself is a constant stretch topological spanner for the underlying quasi unit disk graph and can be extended by means of algorithm rPDT in order to also guarantee constant Euclidean stretch properties. However, this graph is not planar. Applications relying on planar topologies can be executed on the planar backbone and only have to keep track of the *last mile*, the possibly non-planar last-hop connection from the backbone to non-backbone nodes. ReactiveBackbone is the very first reactive topology control algorithm under quasi unit disk model assumptions and, in fact, the first that is explicitly designed for local topology construction. Although it is Ω-reactive and outperforms any beacon-based counterpart, the communication overhead compared to rPDT is significant. Whereas rPDT involves only qualified neighbors from a node's one-hop neighborhood, ReactiveBackbone involves nodes that are at most $k = 8$ hops apart from the executing node, some of which may not even be of further relevance for the constructed topology.

Although it is not reactive, algorithm AsyncPDT introduced in Chapter 9 can be considered a first step towards reactive topology control under relaxed model assumptions. In contrast to the other approaches introduced in this work, this algorithm does not require any node synchronization, nor does it assume that messages are transmitted instantly and reliably. It only presupposes that messages sent over the wireless channel arrive within finite time, possibly by means of several retransmissions. This algorithm also uses PDT for planarization. The output graph is a connected and planar overlay graph of the underlying QUDG and facilitates guaranteed delivery geographic routing. Furthermore, it is shown that for the class of civilized graphs, AsyncPDT can be transformed into a local algorithm, which can be used for local topology control, and therefore, is a good starting point for reactive algorithms research in asynchronous systems.

In summary, regarding the chosen network models, Objective 2 is achieved. Nevertheless, the author is profoundly convinced that further investigation of the problem statement, in particular with regard to quasi unit disk model assumptions, may lead to stronger or at least more elegant solutions.

Objective 3: Reactive geographic routing on planar spanners Concerning the number of required transmissions for a single forwarding operation, state-of-the-art reactive geographic routing algorithms are extremely efficient. For example, algorithm *Rotational Sweep* (RS) [15] requires at most three transmissions for next-hop selection and packet forwarding, regardless of whether the packet is routed in *Greedy* or *Recovery mode*. Furthermore, under unit disk model assumptions, the algorithm provides delivery guarantee. Algorithms of this type are much more efficient than conventional, topology-based, local geographic routing algorithms such as FACE routing [36], which require additional exchange of topology information from all neighboring nodes. However, in order to guarantee delivery, in Recovery mode only specific edges may be used for forwarding. In all known reactive geographic routing algorithms that guarantee message delivery [15, 18, 188, 211, 212], these edges belong either to a planar subgraph of the network graph, or to subgraphs that are free of critical, routing-loop producing, edge intersections (e.g., RS-TT in [15]). To ensure the existence of routing paths that are short in terms of number of hops or Euclidean distance, it is important to select only those edges that belong to constant stretch Euclidean or topological spanners. In the aforementioned algorithms this is not necessarily the case since these use edges of graphs whose Euclidean and topological stretch factors are either unknown or even non-constant. This raises the question: Is it possible to design guaranteed delivery reactive geographic routing algorithms that use only edges of constant stretch Euclidean or topological spanners?

Under unit disk model assumptions, this questioned is positively answers in this thesis by showing that such an algorithm already exists. In Chapter 7 it is proven that algorithm RS (using *Sweep Circle* delay, RS-SC) uses exactly those edges for recovery which would be used by FACE routing if applied onto the partial Delaunay triangulation. In combination with the proof that the partial Delaunay triangulation is a constant stretch Euclidean spanner, it immediately follows that RS-SC already matches the objective. In some scenarios, however, this algorithm produces many unnecessary hops. This observation leads to algorithm *RS-Shortcut*. It combines the constant Euclidean spanner guarantees of RS-SC with the benefits of algorithm RS-TT (algorithm RS using *twisting triangle* delay) which tends to produce recovery paths with fewer hops. Routing paths produced by RS-Shortcut use at most as many hops as those produced by RS-SC, skip up to $\Theta(n)$ unnecessary hops, and use only edges of constant stretch Euclidean spanners, while requiring at most eight transmissions per forwarding step. Regarding these theoretical guarantees, RS-Shortcut is currently the most advanced reactive geographic routing protocol. Therefore, this algorithm matches the objective except for the constant topological spanning guarantees. As in the case of reactive topology control, this algorithm is also not guaranteed to be correct under the more general quasi unit disk graph assumptions.

The situation is partially remedied by introduction of algorithm *Reactive-Virtual-Face-Traversal* in Section 8.6. This is a reactive geographic routing algorithm which guarantees message delivery in quasi unit disk graphs and actually routes along edges of a constant stretch topological spanner, namely the routing graph constructed by algorithm ReactiveBackbone. As mentioned before, this graph can be extended to also satisfy constant Euclidean spanning properties. To the best of the author's knowledge, it is the

first reactive geographic routing protocol under this model at all. Unlike the reactive algorithms for unit disk graphs, this algorithm requires each node on the routing path to construct its entire adjacency in the routing graph. On the positive side, the node degree in this routing graph is bounded by a constant. On the negative side, as discussed previously, the costs incurred by construction of this topology are still significant regarding both the worst-case message complexity, as well as the degree of locality. Hence, under quasi unit disk model assumptions the objective is achieved only partially. It remains an open question if under these model assumptions guaranteed delivery, reactive geographic routing without complete topology construction is possible at all.

In summary, Objective 3 is fully achieved under the unit disk model and partially achieved under the quasi unit disk model.

10.2 Further Discussions and Outlook

Next, the algorithmic approaches introduced in this work are further discussed with an emphasis on common limitations, in particular regarding robustness. These discussions are complemented by ideas and directions on how to further improve these algorithms. Moreover, several other future research directions are outlined.

10.2.1 Message Collision Resolution

In general, contention-based approaches suffer from issues arising from message collisions due to (almost) simultaneous timeouts of delay timers. From a practical point of view, delay functions are discrete and messages are not infinitely small. It may always be the case that two or more nodes respond simultaneously to a request which causes a collision at the intended receiver. Longer delay periods can be used to reduce the probability of a collision, but ultimately this cannot be avoided [18].

If there is no means for a receiver to detect a collision, then algorithms that rely on the right ordering of message arrivals for computation of certain geometric structures (e.g. the algorithms presented in this thesis, or those presented in [15, 18, 68, 69, 107]) are doomed to fail. For example, in algorithm rPDT, if not the closest but another node responds first due to a collision of responses by the actual closest neighbors, the requesting node's local view on the planar graph may be incorrect, which may lead to fatal errors in the application.

The situation is substantially different if a receiver is actually able to detect collisions. A common approach to resolve collisions in reactive algorithms is the use of randomization. For example, in GDBF [187, 188] the nodes that caused the collision are requested to resend, but using a random delay slot within a contention window. In GeRaF [27], all nodes having caused the collision are requested to resubmit with probability $1/2$ until eventually only one node has successfully transmitted. As previously pointed out by Rührup et al. [18], this *probabilistic drop out* is not suitable for reactive topology control and recovery algorithms, in particular not for those presented in this work. In these algorithms, it is necessary that all eligible nodes actually transmit their responses in order to guarantee correctness of the output. Consider the following two examples: In rPDT, a

node's response is necessary for other nodes to compute correctly if they are neighbors in the partial Delaunay triangulation. In algorithm ReactiveBackbone the closest node to a cell's center has to transmit first in order to ensure global consistency of the output graph.

A solution to overcome this problem, which is well suited for the algorithms introduced here, is outlined in [18] and further extended in [212]. The delays of nodes participating in the contention are mapped to one of N time slots. If there are multiple nodes in the first occupied slot, a collision is detected at the receiver and it sends a short request for collision resolution, indicating the index of the slot. The nodes that caused the collision are now remapped into N slots, which reduces the collision probability. This process is repeated until a single node successfully transmits in the first occupied slot. Analytical results in [212] for the case of Greedy routing show that $N = 3$ is sufficient for reasonable success in collision resolution.

The generic scheme for incorporating this strategy into the algorithms presented in this work is as follows. A node u that has just started a contention process and detects a collision reacts by immediately sending of a collision resolution request. This request freezes the delay timers of all nodes currently participating in the contention process and induces the collision resolution process outlined above. Once this process has finished, node u requests all nodes to continue the contention process.

Another solution is based on the observation that collisions are more likely in dense networks (measured, e.g., in terms of average number of neighbors per node) than in sparse networks. In particular in dense networks, the neighbors of a node in planar subgraphs such as GG or PDT tend to be very close because it is less likely that a proximity region of a long link is actually empty. Therefore, these neighbors are particular likely candidates to cause collisions [188]. The solution to this problem is to design delay functions as exponential rather than linear functions of distance. Then, nodes that are nearby are less likely to cause collisions, whereas only faraway nodes—that are anyway likely to become passive during the contention—are more likely to cause collisions. This idea has previously been investigated and concluded to be useful in case of beaconless Greedy routing [189, 190]. These results suggest that the investigation of this solution in the context of network planarization is worthwhile.

10.2.2 Dynamics and Mobile Ad Hoc Networks

The reactive approaches introduced in this work, as well as those on which they are based on, assume stable node neighborhoods during the contention process. From a practical point of view this assumption is critical since wireless network graphs are in fact not static. Due to interference, changes in the environment, and node failure, links can (dis)appear over time. This is also the case in networks consisting partially or entirely of mobile nodes, so called *mobile ad hoc networks*.

There is a variety of models for capturing non-static networks as well as the stochastic nature of the wireless channel (see e.g. [3, 88] and references given therein). Clearly, in the long run it is desirable to formulate algorithmic solutions for those more realistic network models. In a first step, however, the problems considered in this thesis should be studied

using some intermediate model, such as the *edge dynamic (quasi) unit disk graph* [72, 221]. This model has the same characteristics as the (quasi) unit disk model, but models the edge set of the network graph as a function of time, such that edges can (dis)appear dynamically at any given point in time. Guan shows [72, 221] that in such graphs it is still possible to route messages locally while guaranteeing message delivery, essentially under the assumption that there exists a sequence of connected spanning subgraphs of the routing graph, each of which is stable for a sufficiently long period of time. This suggests that similar results can be expected regarding reactive georouting. So far, the problem of reactive topology control has not been considered under such model assumptions and therefore, it is an interesting and open research object. Once the problems arising with dynamics have been better understood under this intermediate model, algorithmic solutions for more realistic network models can be targeted. An appropriate starting point is the constant density spanner construction from Kothapalli et al. [222].

10.2.3 Imperfect Positioning

Most of the algorithms introduced and referenced in this work assume perfect positioning of the nodes in the Euclidean space. This is problematic since positioning methods and systems are generally imperfect [223, 224] and typically limited to specific setups. For example, the *Global Positioning System* (GPS) is limited to outdoor scenarios.

In the presence of positioning errors, the algorithms introduced in this work can fail to compute correct outputs, since the delay functions rely on exact node positions. This holds in particular for those algorithms designed for operation in unit disk graphs. On the contrary, instead of modeling physical obstructions and irregularities in transmission ranges of nodes, the quasi unit disk graph model can can also be used to model uncertainties in node locations in unit disk graphs [71]. Nodes are assumed to be connected according to the unit disk model, but to suffer from positioning errors of at most δ.

In this model, a node is guaranteed to have an edge to any other node at a distance of at most $\mathrm{R} - \delta$, whereas it does not share an edge with nodes at a distance larger than $\mathrm{R} + \delta$. The topology control and geographic routing algorithms for QUDGs introduced in this thesis are then already able to cope with positioning failures of at most δ, where $\frac{\mathrm{R}+\delta}{\mathrm{R}-\delta} \leq \sqrt{2}$, for they require that the ratio of the maximum to the minimum communication radius is at most $\sqrt{2}$.

To further enhance the robustness of the proposed algorithms against position failure it should be studied, if and to which extent ideas and solutions from non-reactive algorithms, that cope with inaccuracies or absence of position information, can be taken over. For example, the approaches presented in [225–229], enable geographic routing in the presence of location errors, whereas in [143, 230–232] the problem of local planarization of wireless network graphs in the absence of any positioning system is addressed.

10.2.4 Reducing Energy Consumption

Preserving the energy resources of network nodes in wireless ad hoc and sensor networks is a major research topic and comes in various flaws (for a comprehensive list see [11]). A

key concept to reduce energy consumption of nodes is the operation of their transceivers at low duty cycle, i.e., to switch off transceivers when not required. At least for small transmit powers, *transmit* and *receive modes* of transceivers consume power in the same order of magnitude. Depending on the architecture this also holds for the transceiver's *idle mode* [3].

To operate correctly, the algorithms introduced in this work require that at algorithm start by some node, all neighboring nodes' transceivers are active and in receive mode to facilitate overhearing of messages. As mentioned above, this consumes power as if they were sending. Therefore, reactive algorithms are not per se more energy efficient than their distributed counterparts that make use of full neighborhood information. Nevertheless, depending on the particular algorithm at hand, there are considerable differences. For example, in order for algorithm BFP [18, 107] to operate correctly, all neighboring nodes have to continue listening during the entire contention period since they may need to send protest messages. In algorithm rPDT, however, all neighboring nodes that have already responded with a CTS message, or have terminally concluded that they are no PDT neighbors, can immediately drop out of the contention, turn off their transceivers, and conserve energy resources.

To the best of the author's knowledge, energy consumption of reactive and conventional topology control algorithms have not yet been compared. Considering the close-to-reality simulation results obtained in [149], which clearly show that beaconless georouting is much more energy efficient than beacon-based georouting, it can be claimed that this also holds for reactive topology control operations.

Various examples can be found in the literature where the reactive approach is further combined with energy saving techniques such as duty-cycling, dynamic transmit power adaption, or avoidance of costly retransmissions by using only the most reliable links. The approaches in [26–30] combine the reactive approach and low duty cycled networking. In [24,31–34] (semi-) beaconless algorithms are introduced that make use of dynamic transmit power adjustment. The protocols described in [22, 23, 25, 174] integrate communication over particularly reliable links and beaconless routing. The algorithm introduced in [202] even combines beaconless routing with duty cycling and adjustment of transmit power.

These examples clearly show that the reactive approach can generally be further complemented with well-known energy saving techniques. Application of these ideas to the approaches presented in this work will lead to even more resource efficient solutions.

10.2.5 Getting Rid of Circle Geometry

Mathews and Frey introduce in [124] the *Localized Link Removal and Addition based Planarization algorithm* (LLRAP). This algorithm does not require any assumption regarding nodes' communication ranges. Instead, it correctly outputs a connected, planar subgraph of a given geometric input graph as long as this input graph satisfies *redundancy* and *coexistence* (see Definitions 3.8 & 3.9). They also show that the class of unit disk graphs is a subclass of the graphs satisfying redundancy and coexistence. That is, algorithms like LLRAP are applicable to a more general graph class than UDG,

while replacing the simplistic unit transmission radius assumptions by structural graph properties that simplify mathematical reasoning.

Reactive algorithms for this graph class (or further generalizations of it concerning k-hop connectivity) have not yet been considered. Using the ideas and techniques presented in this work it is, however, possible to transform LLRAP into a reactive local topology control algorithm.

10.3 Final Remarks

This thesis makes novel and significant contributions to the research area of distributed topology control and geographic routing in wireless ad hoc and sensor networks. Prior to this work, reactive local construction of planar spanners and reactive routing thereon was an open research question [14, 15]. Now it is an established fact. Moreover, the former used to be a rather fuzzy research object. Now it is a precise mathematical term enabling fundamental research in this area.

To close this thesis, some remarks on the impact and the lessons learned are made.

Firstly, the partial Delaunay triangulation was presumed not to be a constant stretch spanner [66, 67]. This work proves the opposite. It even turned out to be the most important instrument for algorithm design in this thesis. This shows that it is always worthwhile to question presumed facts in literature.

Secondly, referring to the surveys and concepts presented, this work structures and broadens the scope of reactive algorithms research. It encourages the application of reactive techniques to non-ideal (non-uniform) networks and to other algorithmic problem statements in topology control. It thereby strengthens this field of study.

Thirdly, the way of thinking that *silence is golden* in wireless ad hoc and sensor networks [7, 8] is an inspiring maxim: It forces the algorithm designer to question every single communication step. Allegedly primitive operations, such as gathering a node's neighborhood, should not be taken for granted.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] T. H. Cormen, C. E. Leiserson, and R. R. Rivest, *Introduction to Algorithms*, 1st ed. Cambridge, MA, USA: MIT Press, 1990.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[3] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Chichester, West Sussex, England: John Wiley & Sons, Ltd, 2005.

[4] F. Martincic and L. Schiebert, *Handbook of Sensor Networks: Algorithms and Architectures*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005, ch. Introduction to Wireless Sensor Networking, pp. 1–40.

[5] T. Haenselmann, *Wireless Sensor Networks: Design Principles for Scattered Systems*. Munich, Germany: Oldenbourg Verlag, 2011.

[6] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*. Chichester, West Sussex, England: John Wiley & Sons, Ltd, 2005.

[7] P. Santi and J. Simon, "Silence Is Golden with High Probability: Maintaining a Connected Backbone in Wireless Sensor Networks," in *Wireless Sensor Networks: Proc. of the 1st European Workshop (EWSN 2004), Berlin, Germany, Jan. 2004*, ser. LNCS, H. Karl, A. Wolisz, and A. Willig, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 2920.

[8] C. Fragouli and A. Orlitsky, "Silence is golden and time is money: Power-aware communication for sensor networks," in *Proc. of the 43rd Allerton Conf. on Communication Control and Computing*, Monticello, IL, USA, Sep. 2005, pp. 1026–1035.

[9] Y. Liu, G. Zhou, J. Zhao, G. Dai, X.-Y. Li, M. Gu, H. Ma, L. Mo, Y. He, J. Wang, M. Li, K. Liu, W. Dong, and W. Xi, "Long-term large-scale sensing in the forest: recent advances and future directions of GreenOrbs," *Frontiers of Computer Science in China*, vol. 4, no. 3, pp. 334–338, Aug. 2010.

[10] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for Smart Dust," in *Proc. of the 5th annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 1999)*, Seattle, WA, USA, Aug. 1999, pp. 271–278.

[11] X. Cui, X. Zhang, and Y. Shang, "Energy-saving Strategies of Wireless Sensor Networks," in *Proc. of the Intl. Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications (MAPE 2007)*, Hangzhou, China, Aug. 2007, pp. 178–181.

[12] M. Heissenbüttel, T. Braun, M. Wälchli, and T. Bernoulli, "Evaluating the limitations of and alternatives in beaconing," *Ad Hoc Networks*, vol. 5, no. 5, pp. 558–578, Jul. 2007.

[13] C. N. Patel, M. S. Shah, and V. N. Pandya, "Comparison Of Reactive Routing Protocols For MANET," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 3, pp. 2–6, 2013.

[14] H. Frey and S. Rührup, "Paving the way towards reactive planar spanner construction in wireless networks," in *Kommunikation in Verteilten Systemen: Proc. of 16. Fachtagung (KiVS 2009), Kassel, Germany, Mar. 2009*, ser. Informatik aktuell, K. David and K. Geihs, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 17–28.

[15] S. Rührup and I. Stojmenović, "Optimizing Communication Overhead while Reducing Path Length in Beaconless Georouting with Guaranteed Delivery for Wireless Sensor Networks," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2440–2453, Dec. 2013.

[16] M. Heissenbüttel and T. Braun, "A Novel Position-based and Beacon-less Routing Algorithm for Mobile Ad-Hoc Networks," in *Proc. of the 3rd IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2003)*, Bern, Switzerland, Jul. 2003, pp. 197–209.

[17] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-based forwarding for mobile ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, Nov. 2003.

[18] S. Rührup, H. Kalosha, A. Nayak, and I. Stojmenović, "Message-Efficient Beaconless Georouting With Guaranteed Delivery in Wireless Sensor, Ad Hoc, and Actuator Networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 95–108, Feb. 2010.

[19] B. Blum, T. He, S. Son, and J. Stankovic, "IGF: A State-Free Robust Communication Protocol for Wireless Sensor Networks," Dept. of Computer Science, University of Virginia, Charlottesville, VA, USA, Tech. Rep. CS-2003-11, 2003. [Online]. Available: http://www.cs.virginia.edu/~th7c/paper/IGF.pdf

[20] D. Chen and P. K. Varshney, "On-demand Geographic Forwarding for data delivery in wireless sensor networks," *Computer Communications*, vol. 30, no. 14–15, pp. 2954–2967, Oct. 2007.

[21] M. Witt and V. Turau, "Robust and low-communication geographic routing for wireless ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 10, no. 4, pp. 486–510, Apr. 2010.

[22] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "BRUMA: Beacon-less Geographic Routing for Multicast Applications," in *Proc. of the IEEE 34th Conf. on Local Computer Networks (LCN 2009)*, Zurich, Switzerland, Oct. 2009, pp. 522–529.

[23] J. A. Sanchez, P. M. Ruiz, and R. Marin-Perez, "Beacon-less Geographic Routing Made Practical: Challenges, Design Guidelines, and Protocols," *IEEE Commun. Mag.*, vol. 47, no. 8, pp. 85–91, Aug. 2009.

[24] C.-H. Feng, Y. Zhang, I. Demirkol, and W. B. Heinzelman, "Stateless Multicast Protocol for Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 2, pp. 240–253, Feb. 2012.

[25] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "Beacon-Less Geographic Routing in Real Wireless Sensor Networks," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 438–450, May 2008.

[26] Y. Zhang and C.-L. Fok, "Receiver-Based Heading: Towards On-Line Energy Efficient Duty Cycle Assignments," in *Proc. of the IEEE Global Communications Conf. (GLOBECOM 2012)*, Anaheim, CA, USA, Dec. 2012, pp. 244–249.

[27] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, Oct./Dec. 2003.

[28] M. C. Vuran and I. F. Akyildiz, "XLP: A Cross-Layer Protocol for Efficient Communication in Wireless Sensor Networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 11, pp. 1578–1591, Nov. 2010.

[29] I. Akyildiz, M. Vuran, and O. Akan, "A Cross-Layer Protocol for Wireless Sensor Networks," in *Proc. of the 40th Annual Conference on Information Sciences and Systems (CISS 2006)*, Princeton, NJ, USA, Mar. 2006, pp. 1102–1107.

[30] P. Casari, M. Nati, C. Petrioli, and M. Zorzi, "Efficient Non-Planar Routing around Dead Ends in Sparse Topologies using Random Forwarding," in *Proc. of the IEEE Intl. Conf. on Communications (ICC 2007)*, Glasgow, UK, Jun. 2007, pp. 3122–3129.

[31] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "Energy-Aware and Time-Critical Geo-Routing in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 4, no. 4, pp. 315–346, 2008.

[32] ——, "PSGR: Priority-based Stateless Geo-Routing in Wireless Sensor Networks," in *Proc. of the 2nd IEEE Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, Washington, DC, USA, Nov. 2005, pp. 673–680.

[33] A. Abdallah, T. Fevens, J. Opatrny, and I. Stojmenović, "Power-aware semi-beaconless 3D georouting algorithms using adjustable transmission ranges for wireless ad hoc and sensor networks," *Ad Hoc Networks*, vol. 8, no. 1, pp. 15–29, Jan. 2010.

[34] S. Das, A. Nayak, S. Rührup, and I. Stojmenović, "Semi-Beaconless Power and Cost Efficient Georouting with Guaranteed Delivery using Variable Transmission Radii for Wireless Sensor Networks," in *Proc. of the 4th IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS 2007)*, Pisa, Italy, Oct. 2007, pp. 1–6.

[35] E. Kranakis, H. Singh, and J. Urrutia, "Compass Routing on Geometric Networks," in *Proc. of the 11th Canadian Conf. on Computational Geometry (CCCG 1999)*, Vancouver, BC, Canada, Aug. 1999, pp. 51–54.

[36] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *Wireless Netw*, vol. 7, no. 6, pp. 609–616, Nov. 2001.

[37] B. Karp and H. T. Kung, "GPSR : Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of the 6th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, USA, Aug. 2000, pp. 243–254.

[38] F. Kuhn, R. Wattenhofer, and A. Zollinger, "An Algorithmic Approach to Geographic Routing in Ad Hoc and Sensor Networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 51–62, Feb. 2008.

[39] H. Frey, F. Ingelrest, and D. Simplot-Ryl, "Localized Minimum Spanning Tree Based Multicast Routing with Energy-Efficient Guaranteed Delivery in Ad Hoc and Sensor Networks," in *Proc. of the 9th IEEE Intl. Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2008)*, Newport Beach, CA, USA, Jun. 2008, pp. 1–8.

[40] J. A. Sanchez, P. M. Ruiz, and I. Stojmenović, "GMR: Geographic Multicast Routing for Wireless Sensor Networks," in *Proc. of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON 2006)*, vol. 1, Reston, VA, USA, Sep. 2006, pp. 20–29.

[41] J. A. Sanchez and P. Ruiz, "LEMA: Localized Energy-Efficient Multicast Algorithm based on Geographic Routing," in *Proc. of the IEEE 31st Conf. on Local Computer Networks (LCN 2006)*, Tampa, FL, USA, Nov. 2006, pp. 3–12.

[42] J. A. Sanchez, P. M. Ruiz, J. Liu, and I. Stojmenović, "Bandwidth-Efficient Geographic Multicast Routing Protocol for Wireless Sensor Networks," *IEEE Sensors J.*, vol. 7, no. 5, pp. 627–636, May 2007.

[43] I. Stojmenović, "Geocasting with Guaranteed Delivery in Sensor Networks," *IEEE Wireless Commun. Mag.*, vol. 11, no. 6, pp. 29–37, Dec. 2004.

[44] N. Mitton, D. Simplot-Ryl, and I. Stojmenović, "Guaranteed Delivery for Geographical Anycasting in Wireless Multi-Sink Sensor and Sensor-Actor Networks," in *Proc. of the 28th IEEE Intl. Conf. on Computer Communications (INFOCOM 2009)*, Rio de Janeiro, Brasil, Apr. 2009, pp. 2691–2695.

[45] Q. Huang, C. Lu, and G.-C. Roman, "Reliable mobicast via face-aware routing," in *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, vol. 3, Hong Kong, Mar. 2004, pp. 2108–2118.

[46] M. Seddigh, J. S. González, and I. Stojmenović, "RNG and internal node based broadcasting algorithms for wireless one-to-one networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 2, pp. 37–44, Apr. 2001.

[47] I. Stojmenović, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 14–25, Jan. 2002.

[48] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," in *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, vol. 4, Mar. 2004, pp. 2458–2468.

[49] X.-Y. Li and I. Stojmenović, "Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Scatternet Formation," in *Proc. of the 1st Intl. Conf. on Ad Hoc Networks and Wireless (ADHOC-NOW 2002)*, Toronto, ON, Canada, Sep. 2002, pp. 17–32.

[50] X.-Y. Li, I. Stojmenović, and Y. Wang, "Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Scatternet Formation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 4, pp. 350–361, Apr. 2004.

[51] H. Frey, "Geographical Cluster Based Multihop Ad Hoc Network Routing with Guaranteed Delivery," in *Proc. of the 2nd IEEE Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, Washington, DC, USA, Nov. 2005, pp. 510–519.

[52] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," *Mobile Netw Appl*, vol. 8, no. 4, pp. 427–442, Aug. 2003.

[53] Y. Deng and I. Stojmenović, "Partial Delaunay Triangulations Based Data-Centric Storage and Routing with Guaranteed Delivery in Wireless Ad Hoc and Sensor Networks," in *Proc. of the Mexican Intl. Conf. on Computer Science (ENC 2009)*, Mexico City, Mexico, Sep. 2009, pp. 24–32.

[54] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, "Mobile object tracking in wireless sensor networks," *Computer Communications*, vol. 30, no. 8, pp. 1811–1825, Jun. 2007.

[55] X. Li, Y. Deng, V. Narasimhan, A. Nayak, and I. Stojmenović, "Localized Address Autoconfiguration in Wireless Ad Hoc Networks," in *Proc. of the Intl. Conf. on Wireless Communications & Signal Processing (WCSP 2010)*, Suzhou, China, Oct. 2010, pp. 1–6.

[56] J. Tan, *Wireless Sensor Networks and Applications*, ser. Signals and Communication Technology. Boston, MA, USA: Springer US, 2008, ch. A Scalable Graph Model and Coordination Algorithms for Mobile Sensor Networks, pp. 65–83.

[57] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with Guaranteed Delivery in ad hoc Wireless Networks," in *Proc. of the 3rd Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 1999)*, Seattle, WA, USA, Aug. 1999, pp. 48–55.

[58] G. Narasimhan and Michiel Smid, *Geometric Spanner Networks*. Cambridge University Press, 2007.

[59] X.-Y. Li, "Topology Control in Wireless Ad Hoc Networks," in *Mobile Ad Hoc Networking*, 1st ed., S. Basagni, M. Conti, S. Giordano, and I. Stojmenović, Eds. Hoboken, NJ, USA: John Wiley & Sons, 2004, pp. 175–203.

[60] J. Suomela, "Survey of Local Algorithms," *ACM Computing Surveys*, vol. 45, no. 2, pp. 1–40, Feb. 2013.

[61] P. Bose and Michiel Smid, "On plane geometric spanners: A survey and open problems," *Computational Geometry*, vol. 46, no. 7, pp. 818–830, Oct. 2013.

[62] P. Xu, Z. Chen, X. Deng, and J. Yu, "A Partial Unit Delaunay graph with Planar and Spanner for Ad hoc Wireless Networks," *Advanced Materials Research*, vol. 267, pp. 322–327, Jun. 2011.

[63] F. Araújo and L. Rodrigues, "Fast Localized Delaunay Triangulation," in *Principles of Distributed Systems: Proc. of the 8th Intl. Conf. (OPODIS 2004), Grenoble, France, Dec. 2004*, ser. LNCS, T. Higashino, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3544, pp. 81–93.

[64] S. Rührup, "Theory and practice of geographic routing," in *Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*, H. Liu, X. Chu, and Y.-W. Leung, Eds. Bentham Science, 2009, pp. 69–88.

[65] X.-Y. Li, "Applications of Computational Geometry in Wireless Networks." in *Ad Hoc Wireless Networking*, X. Cheng, X. Huang, and D.-Z. Du, Eds. Boston, MA, USA: Kluwer Academic Publishers, 2004, pp. 197–264.

[66] X.-Y. Li, G. Calinescu, W. Peng-Jun, and W. Yu, "Localized Delaunay Triangulation with Application in Ad Hoc Wireless Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 10, pp. 1035–1047, Oct. 2003.

[67] X. Li, N. Mitton, I. Simplot-Ryl, and D. Simplot-Ryl, "A Novel Family of Geometric Planar Graphs for Wireless Ad Hoc Networks," in *Proc. of the 30th IEEE Intl. Conf. on Computer Communications (INFOCOM 2011)*, Shanghai, China, Apr. 2011, pp. 1934–1942.

[68] S. Rührup and I. Stojmenović, "Contention-based Georouting with Guaranteed Delivery, Minimal Communication Overhead, and Shorter Paths in Wireless Sensor Networks," in *Proc. of the 24th IEEE Intl. Parallel and Distributed Processing Symposium (IPDPS 2010)*, Atlanta, GA, USA, Apr. 2010, pp. 1–9.

[69] ——, "A New Traversal Scheme for Georouting and Boundary Detection in WSNs," University of Ottawa, SITE, Ottawa, ON, Canada, Tech. Rep. TR-2010-05, Aug. 2010.

[70] K. M. Lillis, S. V. Pemmaraju, and I. A. Pirwani, "Topology Control and Geographic Routing in Realistic Wireless Networks," in *Ad-Hoc, Mobile, and Wireless Networks: Proc. of the 6th Intl. Conf. (ADHOC-NOW 2007), Morelia, Mexico, Sep. 2007*, ser. LNCS, E. Kranakis and J. Opatrny, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4686, pp. 15–31.

[71] ——, "Topology Control and Geographic Routing in Realistic Wireless Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 6, no. 3-4, pp. 265–297, Aug. 2008.

[72] X. Guan, "Better Face Routing Protocols," in *Algorithmic Aspects of Wireless Sensor Networks: Proc. of the 5th Intl. Workshop (ALGOSENSORS 2009), Rhodes, Greece, Jul. 2009*, ser. LNCS, S. Dolev, Ed., vol. 5804. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 167–178.

[73] ——, "Face Routing with Guaranteed Message Delivery in Wireless Ad-hoc Networks," PhD Thesis, University of Toronto, ON, Canada, 2010. [Online]. Available: https://tspace.library.utoronto.ca/handle/1807/19190

[74] F. Neumann and H. Frey, "On the Spanning Ratio of Partial Delaunay Triangulation," in *Proc. of the 9th IEEE Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2012)*, LV, NV, USA, Oct. 2012, pp. 434–442.

[75] M. Benter, F. Neumann, and H. Frey, "Reactive Planar Spanner Construction in Wireless Ad Hoc and Sensor Networks," in *Proc. of the 32nd IEEE Intl. Conf. on Computer Communications (INFOCOM 2013)*, Torino, Italy, Apr. 2013, pp. 2193–2201.

[76] F. Neumann and H. Frey, "Path Properties and Improvements of Sweep Circle Traversals," in *Proc. of the IEEE 9th Intl. Conf. on Mobile Ad-hoc and Sensor Networks (MSN 2013)*, Dalian, China, Dec. 2013, pp. 101–108.

[77] F. Neumann, C. Botterbusch, and H. Frey, "Probing Message Based Local Optimization of Rotational Sweep Paths," in *Ad-hoc, Mobile, and Wireless Networks: Proc. of the 13th Intl. Conf. on Ad-hoc, Mobile, and Wireless Network (ADHOC-NOW 2014), Benidorm Spain, Jun. 2014*, ser. LNCS, S. Guo, J. Lloret, P. Manzoni, and S. Ruehrup, Eds. Cham: Springer International Publishing, Jun. 2014, vol. 8487, pp. 58–71.

[78] F. Neumann and H. Frey, "Partial delaunay triangulation-based asynchronous planarization of quasi unit disk graphs," in *Proc. of the 2nd Intl. Conf. on Networked Systems (NetSys 2015)*, Cottbus, Germany, Mar. 2015, pp. 1–8.

[79] ——, "Foundation of Reactive Local Topology Control," *IEEE Commun. Lett.*, vol. 19, no. 7, pp. 1213–1216, Jul. 2015.

[80] ——, "On Demand Beaconless Planar Backbone Construction for Quasi Unit Disk Graphs," in *Proc. of the 12th Intl. Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2015)*, Dallas, TX, USA, Oct. 2015, pp. 342–351.

[81] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*, vol. 86, no. 1–3, pp. 165–177, Dec. 1990.

[82] S. Schmid and R. Wattenhofer, "Algorithmic Models for Sensor Networks," in *Proc. of the 20th IEEE Intl. Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, Apr. 2006, pp. 1–11.

[83] L. Barrière, P. Fraigniaud, and L. Narayanan, "Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges," in *Proc. of the 5th Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2001)*, Rome, Italy, Jul. 2001, pp. 19–27.

[84] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Ad-Hoc Networks Beyond Unit Disk Graphs," in *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing (DIAL-M-POMC 2003)*, San Diego, CA, USA, Sep. 2003, pp. 69–78.

[85] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks," *ACM Computing Surveys*, vol. 37, no. 2, pp. 164–194, Jun. 2005.

[86] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Ad hoc networks beyond unit disk graphs," *Wireless Netw*, vol. 14, no. 5, pp. 715–729, Jul. 2008.

[87] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Nov. 2000.

[88] R. Wattenhofer, "Sensor Networks: Distributed Algorithms Reloaded – or Revolutions?" in *Proc. of the 13th Colloquium on Structural Information and Communication Complexity (SIROCCO 2006)*, Chester, UK, Jul. 2006, pp. 24–28.

[89] I. A. Kanj and L. Perkovic, "Local Construction of Near-Optimal Power Spanners for Wireless Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 4, pp. 460–474, Apr. 2009.

[90] I. Stojmenović, "Position-Based Routing in Ad Hoc Networks," *IEEE Commun. Mag.*, vol. 40, no. 7, pp. 128–134, Jul. 2002.

[91] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni, and D. Turgut, "Routing protocols in ad hoc networks: A survey," *Computer Networks*, vol. 55, no. 13, pp. 3032–3080, Sep. 2011.

[92] M. Mauve, J. Widmer, and H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad Hoc Networks," *IEEE Netw.*, vol. 15, no. 6, pp. 30–39, Nov. 2001.

[93] A. M. Popescu, I. G. Tudorache, B. Peng, and A. Kemp, "Surveying Position Based Routing Protocols for Wireless Sensor and Ad-hoc Networks," *International Journal of Communication Networks & Information Security*, vol. 4, no. 1, pp. 41–67, Apr. 2012.

[94] G. Finn, "Routing and Addressing Problems in Large Metropolitan-Scale Internetworks," University of Southern California, Tech. Rep. ISI/RR-87-180, Mar. 1987. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a180187.pdf

[95] I. Stojmenović, "Loop-Free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1023–1032, Oct. 2001.

[96] D. Chen and P. K. Varshney, "A Survey of Void Handling Techniques for Geographic Routing in Wireless Networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 1, pp. 50–67, 2007.

[97] J. Bondy and U. Murty, *Graph Theory*, ser. Graduate Texts in Mathematics. Springer, 2008.

[98] H. Frey and I. Stojmenović, "On Delivery Guarantees and Worst-Case Forwarding Bounds of Elementary Face Routing Components in Ad Hoc and Sensor Networks," *IEEE Trans. Comput.*, vol. 59, no. 9, pp. 1224–1238, Sep. 2010.

[99] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. Chichester: John Wiley & Sons, 2000.

[100] Ø. Hjelle and M. Dæhlen, *Triangulations and Applications*. Berlin: Springer, 2006.

[101] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay Graphs Are Almost as Good as Complete Graphs," *Discrete & Computational Geometry*, vol. 5, no. 4, pp. 399–407, Aug. 1990.

[102] J. M. Keil and C. A. Gutwin, "Classes of Graphs Which Approximate the Complete Euclidean Graph," *Discrete & Computational Geometry*, vol. 7, no. 1, pp. 13–28, Jan. 1992.

[103] G. Xia, "The Stretch Factor of the Delaunay Triangulation Is Less than 1.998," *SIAM J. Comput.*, vol. 42, no. 4, pp. 1620–1659, 2013.

[104] X.-Y. Li, G. Calinescu, and P.-J. Wan, "Distributed Construction of a Planar Spanner and Routing For Ad Hoc Wireless Networks," in *Proc. of the 21st Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2002)*, vol. 3, New York City, NY, USA, Jun. 2002, pp. 1268–1277.

[105] K. R. Gabriel and R. R. Sokal, "A New Statistical Approach to Geographic Variation Analysis," *Systematic Zoology*, vol. 18, no. 3, pp. 259–278, Sep. 1969.

[106] G. T. Toussaint, "The Relative Neighbourhood Graph Of A Finite Planar Set," *Pattern Recognition*, vol. 12, no. 4, pp. 261–268, 1980.

[107] H. Kalosha, A. Nayak, S. Rührup, and I. Stojmenović, "Select-and-Protest-Based Beaconless Georouting with Guaranteed Delivery in Wireless Sensor Networks," in *Proc. of the 27th Annual IEEE Intl. Conf. on Computer Communications (INFOCOM 2008)*, Phoenix, AZ, USA, Apr. 2008, pp. 995–1003.

[108] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer, 1985.

[109] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, "On the Spanning Ratio of Gabriel Graphs and beta-Skeletons," *SIAM J. Discrete Math.*, vol. 20, no. 2, pp. 412–427, 2006.

[110] C. Scheideler, "Overlay Networks for Wireless Systems," in *Performance Analysis of Mobile Ad Hoc Networks*, ser. Wireless Networks and Mobile Computing, C. Yu, C. R. Das, and Y. Pan, Eds. Hauppauge, NY, USA: Nova Science Publishers, 2007, vol. 7, ch. 12, pp. 263–301.

[111] P. Boone, E. Chávez, L. Gleitzky, E. Kranakis, J. Opatrny, G. Salazar, and J. Urrutia, "Morelia Test: Improving the Efficiency of the Gabriel Test and Face Routing in Ad-Hoc Networks," in *Structural Information and Communication Complexity: 11th Intl. Colloquium (SIROCCO 2004), Smolenice Castle, Slowakia, Jun. 2004*, ser. LNCS, R. Královič and O. Sýkora, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 3104, pp. 23–34.

[112] X.-Y. Li, "Approximate MST for UDG locally," in *Proc. of the 9th Annual Intl. Conf. on Computing and Combinatorics (COCOON 2003)*, Big Sky, MT, USA, Jul. 2003, pp. 364–373.

[113] J. C. Hou, N. Li, and I. Stojmenović, "Topology Construction and Maintenance in Wireless Sensor Networks," in *Handbook of Sensor Networks: Algorithms and Architectures*, 1st ed., I. Stojmenović, Ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005, pp. 311–341.

[114] P. Bose, P. Carmi, M. Smid, and D. Xu, "Communication-Efficient Construction of the Plane Localized Delaunay Graph," in *Theoretical Informatics: Proc. of the 9th Latin American Symposium (LATIN 2010), Oaxaca, Mexico, Apr. 2010*, ser. LNCS, A. López-Ortiz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6034, pp. 282–293.

[115] L. Luan, W.-J. Hsu, and R. Zhang, "Power-Efficient Geographic Routing for MANETs," *Journal of Information Science and Engineering*, vol. 20, no. 1, pp. 157–180, Jan. 2004.

[116] D. Satyanarayana and S. V. Rao, "Constrained Delaunay Triangulation for Ad Hoc Networks," *Journal of Computer Systems, Networks, and Communications*, vol. 2008, pp. 1–10, 2008.

[117] I. A. Kanj and L. Perković, "Improved Stretch Factor for Bounded-Degree Planar Power Spanners of Wireless Ad-Hoc Networks," in *Algorithmic Aspects of Wireless Sensor Networks: 2nd Intl. Workshop (ALGOSENSORS 2006), Venice, Italy, Jul. 2006*, ser. LNCS, S. E. Nikoletseas and J. D. P. Rolim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4240, pp. 95–106.

[118] I. A. Kanj, L. Perković, and G. Xia, "Strictly-Localized Construction of Near-Optimal Power Spanners for Wireless Ad-Hoc Networks," in *Proc. of the DIALM-POMC Intl. Workshop on Foundations of Mobile Computing (DIAL-M-POMC 2007)*, Portland, OR, USA, Aug. 2007, pp. 1–6.

[119] I. A. Kanj and L. Perković, "On Geometric Spanners of Euclidean and Unit Disk Graphs," in *Proc. of the 25th Intl. Symposium on Theoretical Aspects of Computer Science (STACS 2008), Bordeaux, France*, S. Albers and P. Weil, Eds. Wadern: IBFI Schloss Dagstuhl, Feb. 2008, pp. 409–420.

[120] I. A. Kanj, L. Perković, and G. Xia, "On Spanners and Lightweight Spanners of Geometric Graphs," *SIAM J. Comput.*, vol. 39, no. 6, pp. 2132–2161, 2010.

[121] ——, "Computing Lightweight Spanners Locally," in *Distributed Computing: Proc. of the 22nd Intl. Symposium (DISC 2008), Arcachon, France, Sep. 2008*, ser. LNCS, G. Taubenfeld, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5218, pp. 365–378.

[122] I. A. Kanj and G. Xia, "Improved Local Algorithms for Spanner Construction," in *Algorithms for Sensor Systems: Proc. of the 6th Intl. Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities (ALGOSENSORS 2010), Bordeaux, France, Jul. 2010*, ser. LNCS, C. Scheideler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6451, pp. 1–15.

[123] ——, "Improved local algorithms for spanner construction," *Theoretical Computer Science*, vol. 453, pp. 54–64, Sep. 2012.

[124] E. Mathews and H. Frey, "A Localized Link Removal and Addition based Planarization Algorithm," in *Distributed Computing and Networking: Proc. of the 13th Intl. Conf. (ICDCN 2012), Hong Kong, China, Jan. 2012*, ser. LNCS, L. Bononi, A. K. Datta, S. Devismes, and A. Misra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7129, pp. 337–350.

[125] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric Spanners for Routing in Mobile Networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 174–185, Jan. 2005.

[126] ——, "Geometric Spanner for Routing in Mobile Networks," in *Proc. of the 2nd ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA, USA, Oct. 2001, pp. 45–55.

[127] ——, "Discrete Mobile Centers," in *Proc. of the 17th Annual ACM Symposium on Computational Geometry (SoCG 2001)*, Medford, MA, USA, Jun. 2001, pp. 188–196.

[128] ——, "Discrete Mobile Centers," *Discrete & Computational Geometry*, vol. 30, no. 1, pp. 45–63, May 2003.

[129] K. M. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder, "Geometric Spanners for Wireless Ad Hoc Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 4, pp. 408–421, Apr. 2003.

[130] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," in *Proc. of the 3rd ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, Lausanne, Switzerland, Jun. 2002, pp. 157–164.

[131] Y. Wang and X.-Y. Li, "Localized Construction of Bounded Degree and Planar Spanner for Wireless Ad Hoc Networks," in *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing (DIAL-M-POMC 2003)*, San Diego, CA, USA, Sep. 2003, pp. 59–68.

[132] ——, "Localized Construction of Bounded Degree and Planar Spanner for Wireless Ad Hoc Networks," *Mobile Netw Appl*, vol. 11, no. 2, pp. 161–175, Apr. 2006.

[133] A. C.-C. Yao, "On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems," *SIAM J. Comput.*, vol. 11, no. 4, pp. 721–736, 1982.

[134] W.-Z. Song, Y. Wang, X.-Y. Li, and O. Frieder, "Localized Algorithms for Energy Efficient Topology in Wireless Ad Hoc Networks," in *Proc. of the 5th ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004)*, Tokyo, Japan, May 2004, pp. 98–108.

[135] ——, "Localized Algorithms for Energy Efficient Topology in Wireless Ad Hoc Networks," *Mobile Netw Appl*, vol. 10, no. 6, pp. 911–923, Dec. 2005.

[136] H. Tejeda, E. Chávez, J. A. Sanchez, and P. M. Ruiz, "A Virtual Spanner for Efficient Face Routing in Multihop Wireless Networks," in *Personal Wireless Communications: Proc. of the IFIP TC6 11th Intl. Conf. (PWC 2006), Albacete, Spain, Sep. 2006*, ser. LNCS, P. Cuenca and L. Orozco-Barbosa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4217, pp. 459–470.

[137] ——, "Energy-Efficient Face Routing on the Virtual Spanner," in *Ad-Hoc, Mobile, and Wireless Networks: Proc. of the 5th Intl. Conf. (ADHOC-NOW 2006), Ottawa, Canada, Aug. 2006*, ser. LNCS, T. Kunz and S. S. Ravi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4104, pp. 101–113.

[138] N. Catusse, V. Chepoi, and Y. Vaxès, "Planar Hop Spanners for Unit Disk Graphs," in *Algorithms for Sensor Systems: Proc. of the 6th Intl. Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities (ALGOSENSORS 2010), Bordeaux, France, Jul. 2010*, ser. LNCS, C. Scheideler, Ed., vol. 6451. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 16–30.

[139] M. Damian and S. V. Pemmaraju, "Localized Spanners for Ad Hoc Wireless Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 9, no. 3/4, pp. 305–328, 2010.

[140] N. Li, 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)*, vol. 3, San Franciso, CA, USA, Apr. 2003, pp. 1702–1712.

[141] N. Li, J. C. Hou, and L. Sha, "Design and Analysis of an MST-Based Topology Control Algorithm," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1195–1206, May 2005.

[142] X.-Y. Li, Y. Wang, and W.-Z. Song, "Applications of k-Local MST for Topology Control and Broadcasting in Wireless Ad Hoc Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 12, pp. 1057–1069, Dec. 2004.

[143] R. Wattenhofer and A. Zollinger, "XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks," in *Proc. of the 18th Intl. Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, USA, Apr. 2004, pp. 216–223.

[144] X. Li, N. Mitton, I. Simplot-Ryl, and D. Simplot-Ryl, "Hypocomb: Bounded-Degree Localized Geometric Planar Graphs for Wireless Ad Hoc Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1341–1354, Jul. 2013.

[145] L. Barriére, P. Fraigniaud, L. Narayanan, and J. Opatrny, "Robust position-based routing in wireless ad hoc networks with irregular transmission ranges," *Wireless Communications and Mobile Computing*, vol. 3, no. 2, pp. 141–153, Mar. 2003.

[146] L. Gąsieniec, C. Su, and P. Wong, *Encyclopedia of Algorithms*. Boston, MA, USA: Springer US, 2008, ch. Routing in Geometric Networks, pp. 793–796.

[147] K. Moaveninejad, W.-Z. Song, and X.-Y. Li, "Position-Based Routing for Heterogeneous Wireless Ad Hoc Networks," in *Proc. of the 24th Intl. Conf. on Distributed Computing Systems (ICDCS 2004) Workshops: Intl. Workshop on Wireless Ad Hoc Networking (WWAN 2004)*, Tokyo, Japan, Mar. 2004, pp. 710–715.

[148] ——, "Robust position-based routing for wireless ad hoc networks," *Ad Hoc Networks*, vol. 3, no. 5, pp. 546–559, Sep. 2005.

[149] J. Chen, A. Jiang, I. A. Kanj, G. Xia, and F. Zhang, "Separability and Topology Control of Quasi Unit Disk Graphs," in *Proc. of the 26th Annual IEEE Intl. Conf. on Computer Communications (INFOCOM 2007)*, Anchorage, AK, USA, May 2007, pp. 2225–2233.

[150] ——, "Separability and topology control of quasi unit disk graphs," *Wireless Netw*, vol. 17, no. 1, pp. 53–67, Jan. 2011.

[151] S. Funke and N. Milosavljević, "Guaranteed-Delivery Geographic Routing Under Uncertain Node Locations," in *Proc. of the 26th Annual IEEE Intl. Conf. on Computer Communications (INFOCOM 2007)*, Anchorage, AK, USA, May 2007, pp. 1244–1252.

[152] Q. Fang, J. Gao, L. J. Guibas, V. de Silva, and L. Zhang, "GLIDER: Gradient Landmark-Based Distributed Routing for Sensor Networks," in *Proc. of the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2005)*, vol. 1, Miami, FL, USA, Mar. 2005, pp. 339–350.

[153] S. Funke and N. Milosavljević, "Network sketching or: "How Much Geometry Hides in Connectivity?–Part II"," in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, New Orleans, LA, USA, Jan. 2007, pp. 958–967.

[154] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia, "Local Construction of Planar Spanners in Unit Disk Graphs with Irregular Transmission Ranges," in *Theoretical Informatics: Proc. of the 7th Latin American Symposium (LATIN 2006), Valdivia, Chile, Mar. 2006*, ser. LNCS, J. R. Correa, A. Hevia, and M. Kiwi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 3887, pp. 286–297.

[155] M. Damian and N. Javali, "Distributed construction of low-interference spanners," *Distributed Computing*, vol. 22, no. 1, pp. 15–28, Apr. 2009.

[156] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu, "Greedy Routing with Guaranteed Delivery Using Ricci Flows," in *Proc. of the 8th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN 2009)*, San Francisco, CA, USA, Apr. 2009, pp. 121–132.

[157] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance In Dense Wireless Sensor Networks," in *Proc. of the 1st Intl. Conf. on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, USA, Nov. 2003, pp. 1–13.

[158] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *Proc. of the 1st Intl. Conf. on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, USA, Nov. 2003, pp. 14–27.

[159] T. L. Dinh, D. T. Nguyen, and H. Thuan, "Hybrid Contention-Based Geographic Routing in Wireless Sensor Networks," in *Proc. of the 2nd Intl. Symposium on Information and Communication Technology (SoICT 2011)*, Hanoi, Vietnam, Oct. 2011, pp. 86–91.

[160] L. D. Mendes and J. J.P.C. Rodrigues, "A survey on cross-layer solutions for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 523–534, Mar. 2011.

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

[162] H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals," *IEEE Trans. Commun.*, vol. 32, no. 3, pp. 246–257, Mar. 1984.

[163] T.-C. Hou and V. Li, "Transmission Range Control in Multihop Packet Radio Networks," *IEEE Trans. Commun.*, vol. 34, no. 1, pp. 38–44, Jan. 1986.

[164] G. Chen, T. Sato, and K. Itoh, "Beaconless Location-Based Routing with Signal Strength Assisted for Ad-Hoc Networks," in *Proc. of the 66th IEEE Vehicular Technology Conference (VTC Fall 2007)*, Baltimore, MD, USA, Sep. 2007, pp. 86–90.

[165] Y. Inoue, K. Endo, and Y. Takahashi, "A distance-aware forwarding protocol for beaconless communication in mobile ad hoc networks and its performance," in *Proc. of the 5th Annual Intl. Wireless Internet Conference (WICON 2010)*, Singapore, Mar. 2010, pp. 1–8.

[166] Y. Cao and S. Xie, "A Position Based Beaconless Routing Algorithm for Mobile Ad hoc Networks," in *Proc. of the 3rd Intl. Conf. on Communications, Circuits and Systems (ICCCAS 2005)*, vol. 1, Hong Kong, China, May 2005, pp. 303–307.

[167] M. Watanabe and H. Higaki, "No-Beacon GEDIR: Location-Based Ad-Hoc Routing with Less Communication Overhead," in *Proc. of the 4th Intl. Conf. on Information Technology (ITNG 2007)*, Las Vegas, NV, USA, Apr. 2007, pp. 48–55.

[168] H. Füßler, J. Widmer, M. Mauve, and H. Hartenstein, "A Novel Forwarding Paradigm for Position-Based Routing (with Implicit Addressing)," in *Proc. of the IEEE Annual Computer Communications Workshop (CCW 2003)*, Dana Point, CA, USA, Oct. 2003, pp. 194–200.

[169] X. Shi and K. Liu, "A Contention-Based Beaconless Geographic Routing Protocol for Mobile Ad Hoc Networks," in *Proc. of the 3rd Intl. Conf. on Communications and Networking in China (ChinaCom 2008)*, Hangzhou, China, Aug. 2008, pp. 840–843.

[170] M. Witt and V. Turau, "BGR: Blind Geographic Routing for Sensor Networks," in *Proc. of the 3rd Intl. Workshop on Intelligent Solutions in Embedded Systems (WISES 2005)*, Hamburg, Germany, May 2005, pp. 51–61.

[171] T. Lee, C. Qiao, M. Demirbas, and J. Xu, "ABC: A Simple Geographic Forwarding Scheme Capable of Bypassing Routing Holes in Sensor Networks," *Ad Hoc Networks*, vol. 8, no. 4, pp. 361–377, Jun. 2010.

[172] I. Amadou and F. Valois, "Pizza Forwarding: A Beaconless Routing Protocol Designed for Realistic Radio Assumptions," in *Proc. of the 4th Intl. Conf. on Sensor Technologies and Applications (SENSORCOMM 2010)*, Venice/Mestre, Italy, Jul. 2010, pp. 495–500.

[173] I. Amadou, G. Chelius, and F. Valois, "Energy-Efficient Beacon-less Protocol for WSN," in *Proc. of the 22nd Annual IEEE Intl. Symposium on Personal Indoor and Mobile Radio Communications (PIMRC 2011)*, Toronto, ON, Canada, Sep. 2011, pp. 990–994.

[174] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "BOSS: Beacon-less On Demand Strategy for Geographic Routing in Wireless Sensor Networks," in *Proc. of the 4th IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS 2007)*, Pisa, Italy, Oct. 2007, pp. 1–10.

[175] M. Jain, M. K. Mishra, and M. M. Gore, "Energy Aware Beaconless Geographical Routing in Three Dimensional Wireless Sensor Networks," in *Proc. of the 1st Intl. Conf. on Advanced Computing (ICAC 2009)*, Chennai, India, Dec. 2009, pp. 122–128.

[176] M. K. Mishra and M. M. Gore, "An Improved Forwarder Selection Approach for Energy Aware Geographic Routing in Three Dimensional Wireless Sensor Networks," in *Proc. of the Intl. Conf. on Communication, Computing and Security (ICCCS 2011)*, Odisha, India, Feb. 2011, pp. 166–171.

[177] R. Nawaz, S. A. Hussain, S. A. Abid, and J. Shafi, "Beaconless Multihop Routing Protocol for Wireless Sensor Networks," in *Proc. of the IEEE 3rd Intl. Conf. on Communication Software and Networks (ICCSN 2011)*, Xi'an, China, May 2011, pp. 721–725.

[178] R. Jurdak, A. G. Ruzzelli, G. M. P. O'hare, and R. Higgs, "Directed Broadcast with Overhearing for Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 3, pp. 1–35, Dec. 2009.

[179] Y. Yim, H. Park, J. Lee, S. Oh, and S.-H. Kim, "Distributed Forwarder Selection for Beaconless Real-Time Routing in Wireless Sensor Networks," in *Proc. of the 77th IEEE Vehicular Technology Conference (VTC Spring 2013)*, Dresden, Germany, Jun. 2013, pp. 1–5.

[180] M. Al-Otaibi, H. Soliman, and J. Zheng, "A multipath routeless routing protocol with an efficient location update mechanism," *Int. J. Internet Protocol Technology*, vol. 6, no. 1/2, pp. 75–82, Jun. 2011.

[181] D. Rosário, Z. Zhao, A. Santos, T. Braun, and E. Cerqueira, "A beaconless Opportunistic Routing based on a cross-layer approach for efficient video dissemination in mobile multimedia IoT applications," *Computer Communications*, vol. 45, no. 1, pp. 21–31, Jun. 2014.

[182] H. Hartenstein and K. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Networks," *IEEE Commun. Mag.*, vol. 46, no. 6, pp. 164–171, Jun. 2008.

[183] H. Füßler, H. Hartenstein, M. Mauve, W. Effelsberg, and J. Widmer, "Contention-Based Forwarding for Street Scenarios," in *Proc. of the 1st Intl. Workshop in Intelligent Transportation (WIT 2004)*, Hamburg, Germany, Mar. 2004, pp. 1–5.

[184] Y. Sasaki, W.-C. Lee, T. Hara, and S. Nishio, "On Alleviating Beacon Overhead in Routing Protocols for Urban VANETs," in *Proc. of the 14th IEEE Intl. Conf. on Mobile Data Management (MDM 2013)*, Milan, Italy, Jun. 2013, pp. 66–76.

[185] The Editors of IEEE 802.11, *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, IEEE Std., Nov. 1997.

[186] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LAN's," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 212–225, Oct. 1994.

[187] M. Chawla, N. Goel, K. Kalaichelvan, A. Nayak, and I. Stojmenović, "Beaconless Position Based Routing with Guaranteed Delivery for Wireless Ad-Hoc and Sensor Networks," in *Ad-Hoc Networking: Proc. of the IFIP 19th World Computer Congress (WCC 2006), IFIP TC6 Interactive Conf. on Ad-Hoc Networking, Santiago de Chile, Chile, Aug. 2006*, ser. IFIP International Federation for Information Processing, K. Agha, Ed. Boston, MA, USA: Springer US, 2006, vol. 212, pp. 61–70.

[188] ——, "Beaconless Position-Based Routing with Guaranteed Delivery for Wireless Ad hoc and Sensor Networks," *Acta Automatica Sinica*, vol. 32, no. 6, pp. 846–855, Nov. 2006.

[189] P. Škraba, H. Aghajan, and A. Bahai, "Cross-Layer Optimization for High Density Sensor Networks: Distributed Passive Routing Decisions," in *Ad-Hoc, Mobile, and Wireless Networks: Proc. of the 3rd Intl. Conf. (ADHOC-NOW 2004), Vancouver, BC, Canada, Jul. 2004*, ser. Lecture Notes in Computer Science, I. Nikolaidis, M. Barbeau, and E. Kranakis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 3158, pp. 266–279.

[190] ——, "Distributed Passive Routing Decisions in Mobile Ad-Hoc Networks," in *Proc. of the 60th IEEE Vehicular Technology Conference (VTC Fall 2004)*, vol. 4, Los Angeles, CA, USA, Sep. 2004, pp. 2814–2818.

[191] M. Zorzi and R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 337–348, Oct./Dec. 2003.

[192] T. Aguilar, S.-J. Syue, V. Gauthier, H. Afifi, and C.-L. Wang, "CoopGeo: A Beaconless Geographic Cross-Layer Protocol for Cooperative Wireless Ad Hoc Networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 8, pp. 2554–2565, Aug. 2011.

[193] D. Chen, J. Deng, and P. Varshney, "A State-Free Data Delivery Protocol for Multihop Wireless Sensor Networks," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2005)*, vol. 3, New Orleans, LA, USA, Mar. 2005, pp. 1818–1823.

[194] H. Zhang and H. Shen, "Energy-Efficient Beaconless Geographic Routing in Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 881–896, Jun. 2010.

[195] ——, "EEGR: Energy-Efficient Geographic Routing inWireless Sensor Networks," in *Proc. of the Intl. Conf. on Parallel Processing (ICPP 2007)*, Xi'an, China, Sep. 2007, pp. 67–74.

[196] N. Mitton, E. Natalizio, and R. Wolhuter, "Beacon-Less Mobility Assisted Energy Efficient Georouting in Energy Harvesting Actuator and Sensor Networks," in *Ad-hoc, Mobile, and Wireless Network: Proc. of the 12th Intl. Conf. (ADHOC-NOW 2013), Warsaw, Poland, Jul. 2013*, ser. LNCS, J. Cichoń, M. Gebala, and M. Klonowski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7960, pp. 281–292.

[197] O. Jumira, R. Wolhuter, and S. Zeadally, "Energy-efficient beaconless geographic routing in energy harvested wireless sensor networks," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 1, pp. 58–84, Jan. 2013.

[198] M. Zorzi, "A new contention-based MAC protocol for geographic forwarding in ad hoc and sensor networks," in *Proc. of the IEEE Intl. Conf. on Communications (ICC 2004)*, vol. 6, Paris, France, Jun. 2004, pp. 3481–3485.

[199] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "PSGR: Priority-based Stateless Geo-Routing in Highly Dynamic Sensor Networks," Pennsylvania State University, Tech. Rep. CSE 04-021, 2004. [Online]. Available: http://www.cse.psu.edu/pda/SDB/pubs/PSGR_TR.pdf

[200] C. Huang and G. Wang, "Contention-Based Beaconless Real-Time Routing Protocol for Wireless Sensor Networks," *Wireless Sensor Network*, vol. 2, no. 7, pp. 528–537, 2010.

[201] J. Kuruvila, A. Nayak, and I. Stojmenović, "Progress and Location Based Localized Power Aware Routing for Ad Hoc and Sensor Wireless Networks," *International Journal of Distributed Sensor Networks*, vol. 2, no. 2, pp. 147–159, 2006.

[202] L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo, "A MAC/Routing cross-layer approach to geographic forwarding in wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 6, pp. 872–884, Aug. 2007.

[203] P. Dong, H. Qian, X. Wei, S. Lan, and C. Pu, "A Beacon-Less Geographic Multipath Routing Protocol for Ad Hoc Networks," *Mobile Netw Appl*, vol. 18, no. 4, pp. 500–512, Aug. 2013.

[204] P. M. Ruiz, V. Cabrera, J. A. Martinez, and F. J. Ros, "BRAVE: Beacon-less Routing Algorithm for Vehicular Environments," in *Proc. of the 7th IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS 2010)*, San Francisco, CA, USA, Nov. 2010, pp. 709–714.

[205] M. Asgari, M. Ismail, and R. Alsaqour, "Reliable Contention-based Beaconless Packet Forwarding Algorithm for VANET Streets," *Procedia Technology*, vol. 11, pp. 1011–1017, 2013.

[206] K. Z. Ghafoor, K. Abu Bakar, J. Lloret, R. H. Khokhar, and K. C. Lee, "Intelligent beaconless geographical forwarding for urban vehicular environments," *Wireless Netw*, vol. 19, no. 3, pp. 345–362, Apr. 2013.

[207] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea, "VANET Routing on City Roads Using Real-Time Vehicular Traffic Information," *IEEE Trans. Veh. Technol.*, vol. 58, no. 7, pp. 3609–3626, Sep. 2009.

[208] S. A. Chhoeun, K. S. N. Ayutaya, C. Charnsripinyo, K. Chamnongthai, and P. Kumhom, "A Novel Message Fabrication Detection for Beaconless Routing in VANETs," in *Proc. of the Intl. Conf. on Communication Software and Networks (ICCSN 2009)*, Macau, Feb. 2009, pp. 453–457.

[209] B. Wilson and K. G. Preetha, "Enhanced Beaconless Routing Mechanism in VANET," in *Proc. of the 2nd Intl. Conf. on Advances in Computing, Communications and Informatics (ICACCI 2012)*, Mysore, India, Aug. 2013, pp. 1076–1081.

[210] J. A. Martinez, D. Vigueras, F. J. Ros, and P. M. Ruiz, "Evaluation of the Use of Guard Nodes for Securing the Routing in VANETs," *Journal of Communications and Networks*, vol. 15, no. 2, pp. 122–131, Apr. 2013.

[211] M. Narasawa, M. Ono, and H. Higaki, "NB-FACE: No-Beacon FACE Ad-hoc Routing Protocol for Reduction of Location Acquisition Overhead," in *Proc. of the 7th Intl. Conf. on Mobile Data Management (MDM 2006)*, Nara, Japan, May 2006, pp. 1–5.

[212] P. Kulakowski, E. Egea-López, J. García-Haro, and L. Orozco-Barbosa, "ARROW: Azimuth-Range ROuting for large-scale Wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 93, pp. 1–19, 2013.

[213] H. Park, J. Lee, S. Oh, Y. Yim, and S.-H. Kim, "Distributed Multicast Protocol Based on Beaconless Routing for Wireless Sensor Networks," in *Proc. of the IEEE Intl. Conf. on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, Jan. 2013, pp. 522–523.

[214] L. Galluccio, G. Morabito, and S. Palazzo, "GEographic Multicast (GEM) for Dense Wireless Networks: Protocol Design and Performance Analysis," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1332–1346, Aug. 2013.

[215] F. Aurenhammer and R. Klein, "Voronoi Diagrams," in *Handbook of Computational Geometry*, 1st ed., J.-R. Sack and J. Urrutia, Eds. Amsterdam: North-Holland, 2000, pp. 201–291.

[216] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Berlin. Heidelberg: Springer Berlin Heidelberg, 2000.

[217] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the Shape of a Set of Points in the Plane," *IEEE Trans. Inf. Theory*, vol. 29, no. 4, pp. 551–559, Jul. 1983.

[218] H. Frey and D. Görgen, "Planar graph routing on geographical clusters," *Ad Hoc Networks*, vol. 3, no. 5, pp. 560–574, Sep. 2005.

[219] H. Frey and D. Görgen, "Geographical Cluster-Based Routing in Sensing-Covered Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 9, pp. 899–911, Sep. 2006.

[220] Y. Cohen and J. Opatrny, "A Local Algorithm for Dominating Sets of Quasi-unit Disk Graphs," in *Proc. of the 2nd Canadian Conf. on Computer Science and Software Engineering (C3S2E 2009)*, Montreal, QC, Canada, May 2009, pp. 223–231.

[221] X. Guan, "Face Traversal Routing on Edge Dynamic Graphs," in *Proc. of the 19th Intl. Parallel and Distributed Processing Symposium (IPDPS 2005)*, Denver, CO, USA, Apr. 2005, pp. 1–8.

[222] K. Kothapalli, C. Scheideler, M. Onus, and A. Richa, "Constant Density Spanners for Wireless Ad-Hoc Networks," in *Proc. of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2005)*, Las Vegas, NV, USA, Jul. 2005, pp. 116–125.

[223] S. Slijepcevic, S. Megerian, and M. Potkonjak, "Location Errors in Wireless Embedded Sensor Networks: Sources, Models, and Effects on Applications," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 3, pp. 67–78, Jul. 2002.

[224] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, Aug. 2001.

[225] S. Kwon and N. B. Shroff, "Geographic routing in the presence of location errors," *Computer Networks*, vol. 50, no. 15, pp. 2902–2917, Oct. 2006.

[226] T. Watteyne, I. Augé-Blum, M. Dohler, and D. Barthel, "Geographic Forwarding in Wireless Sensor Networks with Loose Position-Awareness," in *Proc. of the IEEE 18th Intl. Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, Athens, Greece, Sep. 2007, pp. 1–5.

[227] A. M. Popescu, N. Salman, and A. H. Kemp, "Energy Efficient Geographic Routing Robust Against Location Errors," *IEEE Sensors J.*, vol. 14, no. 6, pp. 1944–1951, Jun. 2014.

[228] B. Peng and A. Kemp, "Energy-efficient geographic routing in the presence of localization errors," *Computer Networks*, vol. 55, no. 3, pp. 856–872, Feb. 2011.

[229] S. Subramanian and S. Shakkottai, "Geographic Routing With Limited Information in Sensor Networks," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4506–4519, Sep. 2010.

[230] M. Kadivar, M. Shiri, and M. Dehghan, "Distributed topology control algorithm based on one- and two-hop neighbors' information for ad hoc networks," *Computer Communications*, vol. 32, no. 2, pp. 368–375, Feb. 2009.

[231] M. Dyer, J. Beutel, and L. Thiele, "S-XTC: A Signal-Strength Based Topology Control Algorithm for Sensor Networks," in *Proc. of the 8th Intl. Symposium on Autonomous Decentralized Systems (ISADS 2007)*, Sedona, AZ, USA, Mar. 2007, pp. 508–518.

[232] J.-P. Sheu, S.-C. Tu, and C.-H. Hsu, "Location-free topology control protocol in wireless ad hoc networks," *Computer Communications*, vol. 31, no. 14, pp. 3410–3419, Sep. 2008.