UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Competitive Routing in Hybrid Communication Networks and Message efficient SetCover in Ad Hoc Networks

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
an der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

vorgelegt von

Christina Kolb

Paderborn, Januar 2022

Betreuer:

Prof. Dr. Christian Scheideler	(Universität Paderborn)

Gutachter:

Prof. Dr. Christian Scheideler	(Universität Paderborn)
Prof. Dr. Friedhelm Meyer auf der Heide	(Universität Paderborn)

Weitere Mitglieder der Promotionskommission:

Prof. Dr. Eckhard Steffen	(Universität Paderborn)
Dr. rer. nat. Matthias Fischer	(Universität Paderborn)
Dr. rer. nat. Florian Klingler	(Universität Paderborn)

Datum der mündlichen Prüfung:

9. März 2022

Acknowledgment

First of all, I wish to express special thanks to Prof. Dr. Christian Scheideler for being my thesis supervisor, particularly for his great support through the past years. He gave me freedom and independence for my working style. With his agreement, I attended a wide range of seminars and workshops that helped me to improve my research thinking and writing style. Thank you so much for this interesting research topic on competitive routing in hybrid communication networks with holes. This topic was love at first sight and I could work on it forever.

The research presented in this thesis was supported by the SFB 901. I am grateful for the funding I received. Thanks a lot to Prof. Dr. Mariëlle Stoelinga who gave me the time to finish this thesis besides working on my new research area on safety and security analysis. Heartful thanks to the Mentoring Program 2016/17. The workshops and especially the meetings with other PhD students encouraged me a lot to continue with researching at a postdoc position after the PhD years.

Thanks to all my co-authors Dr. Daniel Jung, Jannik Castenow, and Thorsten Götte, for the fruitful discussions. Working and writing with you together contributed a lot to the research world. Thanks also to Dr. Lea Budde, Hans Kolb, Jannik Castenow, and Thorsten Götte, for the proof reading of this thesis. Many thanks to Alina Ergova and Jannik Castenow for designing the clearest and most beautiful graphics for some of my conference talks. Some of them are also included in this thesis. I wish to thank my colleagues in the research group for the very friendly and creative atmosphere. Our research-group-events and also the game evenings were fun!

I am also thankful to my friends for being able to spend a rejuvenating free time to obtain fresh motivation and energy for this thesis, especially Filmcrew Paderborn and my new friends from mentoring programs. Thanks to all my running friends and events. Walking through the highs and lows of life together with you was incredibly inspiring for my life and thus for this thesis!

Last but not least, I would particular like to thank Paula, Paultje, and Alex. This thesis would not have been completed without your love and encouragement.

Christina Kolb

Zusammenfassung

Routing ist in drahtlosen Ad-Hoc Netzwerken eine große Herausforderung, vor allem, wenn Knoten mobil sind und so weit auseinander liegen, dass entweder viele Hops oder lange Wege benötigt werden, um eine Nachricht von einem Knoten zu einem anderen zu senden. In der Tat ist bekannt, dass jedes Online-Routing-Protokoll im worst-case sehr schlecht performt, genau in dem Sinne, dass es eine Verteilung der Knoten gibt, die zu einem schlechten Routingpfad für das Protokoll führt, sogar dann, wenn Knoten zwar ihre geographische Position kennen, aber die des Nachrichtenziels unbekannt ist. Der Grund hierfür sind Funklöcher im Ad-Hoc Netzwerk: Diese erfordern große Routingumwege bis eine Nachricht ihr Ziel erreicht und die nur sehr schwer mit einem Online-Routing-Ansatz zu finden sind.

Die Annahme in dieser Doktorarbeit ist ein schnurloses Ad-Hoc Netzwerk, das begrenzt long-range-links nutzen kann, welche über eine globale Kommunikationsinfrastruktur, wie zum Beispiel eine zellulare Infrastruktur oder einen Satelliten, zur Verfügung gestellt wird. Das Ziel ist hierdurch eine Abstraktion des drahtlosen Ad-Hoc Netzwerks zu berechnen, um möglichst kurze Routingwege im Ad-Hoc Netzwerk zu finden.

Wir präsentieren einen verteilten Algorithmus, der eine Abstraktion des Ad-Hoc Netzwerk in $\mathcal{O}(\log^2 n)$ Runden mit Hilfe von long-range-links berechnet. Das Resultat sind c-kompetative Routingpfade zwischen Knotenpaaren des Ad-Hoc Netzwerks für eine Konstante c. Wir nehmen in diesem Fall an, dass sich die konvexen Hüllen der Funklöcher nicht überschneiden.

Des Weiteren zeigen wir, dass der dafür benötigte Speicher nur von er Anzahl der Knoten und der Größe des Funklochs abhängt und komplett unabhängig von der Gesamntanzahl der Knoten des Ad-Hoc Netzwerks. Diese Information müssen nur ein paar Knoten kennen, um das Routing erfolgreich zu gewährleisten.

Zusätzlich zu den konvexen Hüllen als Funklochabstraktion, berechnen wir Bounding Boxen als Funklochabstraktion. Der Vorteil von Bounding Boxen ist, dass man nur eine konstante Anzahl der Funklochknoten betrachten braucht. Wir beweisen, dass Bounding Boxen eine sinnvolle Funklochabstraktion sind, um auf c-kompetativen Wegen im Ad-Hoc Netzwerk zu routen, füer den Fall, dass sich die Bounding Boxen nicht üeberschneiden. Für den Fall, dass sich mehrere Bounding Boxen überschneiden, zeigen wir, dass unser Routingansatz der bestmöegliche für eine Online-Routing-Strategie ist. Zusätzlich präsentieren wir eine c-kompetative

Routingstrategie für sich paarweise üeberschneidende Bounding Boxen.

Außerdem präsentieren wir einen Nachrichten-effizienten Algorithmus für das Mengenüberdeckungsproblem, auch SetCover genannt. Hierbei ist eine Grundmenge U gegeben, die n Elemente enthält, und m Teilmengen von U. Das Ziel ist eine minimale Anzahl von diesen Teilmengen zu finden, deren Vereinigung alle Elemente aus U enthält. Ursprünglich hat jede Menge eine bidirektionale Kommunikationsverbindung mit jedem Element, das sie enthält. Unser Resultat ist ein $\tilde{O}(\log^2(\Delta))$-Zeit und $O(\sqrt{\Delta}(n+m))$-Nachrichten Algorithmus mit erwarteter Approximation von $O(\log(\Delta))$ im KT_0 Modell, wobei Δ die maximale Kardinalität von jeder Teilmenge ist. Dieser Algorithmus ist *fast* optimal in Bezug auf die Zeit- und Nachrichtenkomplexität.

Abstract

Competitive routing is a challenging problem for wireless ad hoc networks, for example, when the nodes are mobile, spread so widely, and in the most cases multiple hops and long paths are needed to route a message from a source node to a target. It is known that any online routing algorithm has a poor performance in the worst case, i.e., there is a distribution of nodes resulting in bad routing paths for that algorithm. This happens even if the nodes know their geographic positions and the geographic position of the destination of a message. This is because radio holes in the ad hoc network require messages to route via long detours in order to get to a destination. This is hard to find in an online fashion.

In this thesis, we assume that a wireless ad hoc network makes limited use of long-range links. The long-range links are provided by a global communication infrastructure which can be a cellular infrastructure or a satellite. The global communication infrastructure helps to compute an abstraction of the wireless ad hoc network. The abstraction allows the messages to be sent along short paths in the ad hoc network.

We present distributed algorithms which compute an abstraction of the ad hoc network in $\mathcal{O}\left(\log^2 n\right)$ time with the help of long-range links, which provides c-competitive routing paths between any two nodes of the ad hoc network for some constant c for the case that the convex hulls of the radio holes do not intersect. We show that the storage that is needed for the abstraction is dependent on the number and the size of the radio holes in the wireless ad hoc network. It is independent of the total number of nodes and that information just has to be known to a few nodes for the routing to work.

In addition to convex hulls, we also compute a bounding box abstraction of the radio holes in the ad hoc network. The advantage of using bounding boxes as hole abstraction is that we need a constant number of nodes per hole. We prove that bounding boxes as hole abstraction allows us to find c-competitive paths in the ad hoc network for non-intersecting bounding boxes. For the case of intersecting bounding boxes, we provide a routing strategy and show via simulations that this strategy significantly outperforms the best online routing strategies ever for wireless ad hoc networks. Finally, we present a routing strategy that is c-competitive for pairwise intersecting bounding boxes.

Besides competitive routing in hybrid communication networks, we observe a message-

efficient distributed algorithm for the SetCover Problem in the KT_0 model which one can imagine as the ad hoc network. The SetCover Problem has a given ground set U of n elements and m subsets of U. The goal is to provide the minimal number of these subsets that contain all elements. The default distributed setup of this problem considers that each set has a bidirected communication link with each element it contains. In this thesis, we present a distributed algorithm to solve this problem within $\tilde{O}(\log^2(\Delta))$-time and with a $O(\sqrt{\Delta})(n+m))$-message algorithm that has an expected approximation ration of $O(\log(\Delta))$ in the KT_0 model. Here Δ denotes the maximal cardinality of each subset. This algorithm is *almost* optimal with regard to time and message complexity.

Contents

1 Introduction

1.1 A short story

Imagine the following scenario: My colleagues Jan, Max, Nico, Philip, Angelina, Alex, Robert, and Julia are playing handball, see Figure 1.1.

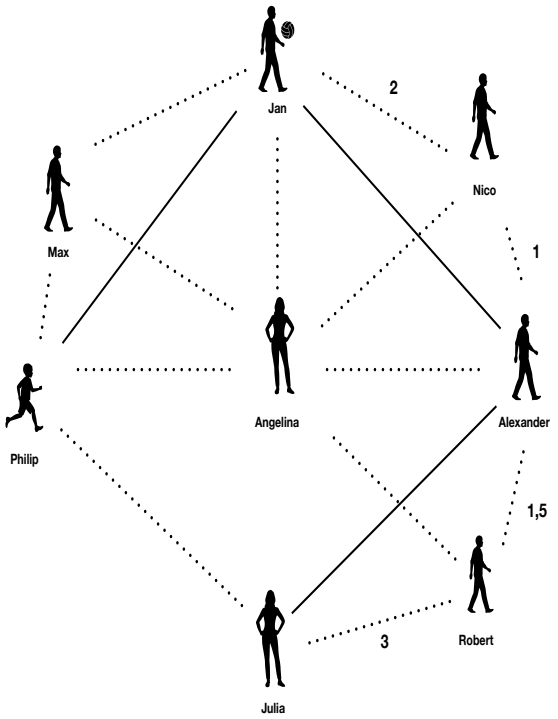

Figure 1.1: Handball match with two different communication modes: The black dashed lines represent communication on short distances, such as talking. The black lines represent the communication via a long distance, such as waving hands.

Of course, in a handball team it is important to communicate properly with other team members during a match. Here, two different communication modes can come into play: In the first communication mode, players can talk to each other. For example, Angelina says to Philip: "I am open. Pass the ball to me!", see the black

dashed lines in the figure. In the second communication mode, players can give each other hand signs. For example, Jan can wave with his hands to Philip to show that he is open and that Philip can pass the ball via Max to Julia. For an illustration see the black lines in the figure.

While the first communication mode only makes sense via short distances, because only players that stand close to each other are also able to hear each other, the second communication mode is useful for larger distances where players are not able to hear each other properly.

1.2 The Hybrid Communication Model

But how is the handball annecdote above related to computer science?

In handball, we are interested in passing the ball from one player to another to reach the goal of the other team. In contrast, computer science we are interested in sending large amounts of data from one device to another. Instead of the black dashed lines for communication on short distances during handball matches, we consider a Wi-Fi mode for electronic devices, where the black arrows in Figure 1.2 represent the black dashed lines for the previous annecdote. The black lines of the communication mode using hand signs can be interpreted as an internet-mode (red edges in Figure 1.2). Just replace each handball player with a smartphone and, voilà, you have an idea about the hybrid communication model that we utilize in this thesis.

To sum up, the ***hybrid communication model*** consists of two different communication networks, namely the ***ad hoc network*** and the ***cellular infrastructure***. The ad hoc network is, for example, a set of connected smartphones, that communicate with each other via ***Wi-Fi***. The Wi-Fi is for free and can be used to transmit a large amount of data, but only on short distances. In contrast to Wi-Fi, using ***internet*** connections involves costs but enables us to communicate directly with a target smartphone via long-range links.

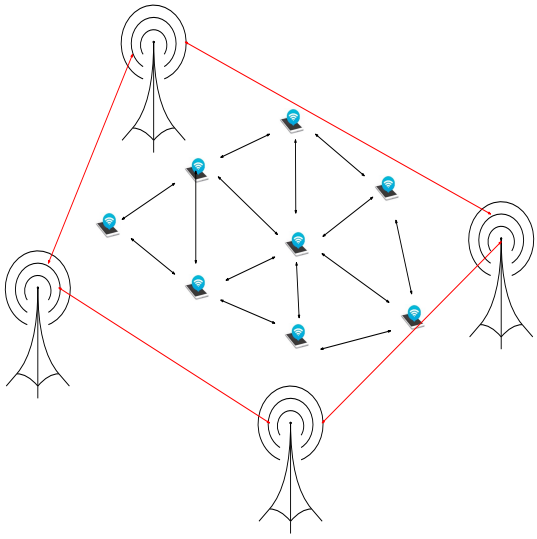

Figure 1.2: Hybrid communication model Wi-Fi connections (black arrows) and the cellular infrastructure (red long-range links).

1.3 Competitive Routing

We now assume in the handball annecdote above that each throw has a certain length. This could be the length of the distance that the ball has to overcome between each pair of players. For example, Jan throws the ball to Nico who is 2 meters far away from Jan. Then Nico passes the ball to Alex who is 1 meter away from Nico, and so on. The length of the path from Jan to Julia is the sum of the distances of each throw which is $2 + 1 + 1.5 + 3 = 7.5$ meters.

We consider competitive paths with respect to the Euclidean distance, i.e., competitive paths between a source and a target are only slightly longer than the shortest distance between source and target.
This also holds for the smartphone setting, where we route data from a source smartphone to a target smartphone over the black Wi-Fi lines such that this path need not necessarily be the optimal shortest path, but its length should be at least very close to the optimal shortest path.

We call a routing path from a source to a target smartphone ***competitive***, if

$$\sum d(u_i, v_j) \leq c \cdot \sum spd(u_i, v_j).$$

In words: the sum of the length of Wi-Fi connections is less than or equal to a constant time of the sum of the length of the Wi-Fi connections of the shortest path.

1.4 Radio Holes and the Problem of Radio Holes

Figure 1.3 presents trees as obstacles on a handball court.

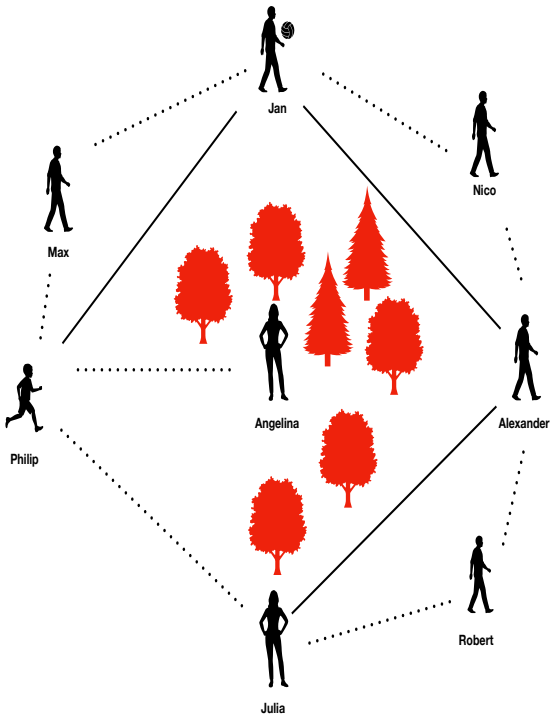

Figure 1.3: Handball match with trees as obstacles.

These trees interfere with the visibility between two players and thus prevent the players from using the two communication modes. For example, the trees hinder the players Max, Jan, Nico, and Alex from talking directly to Angelina or from communicating via hand signs with her. We call this kind of obstacles holes. In the smartphone setting, we refer to this kind of holes as radio holes (i.e., the grey face in Figure 11.1).

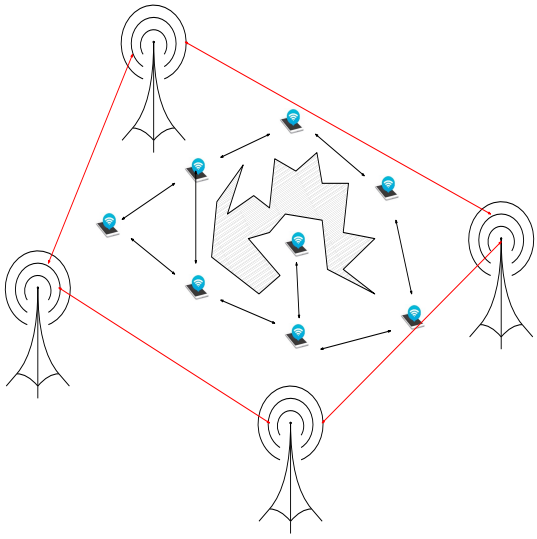

Figure 1.4: Hybrid communication model with radio holes

You can imagine these radio holes as physical objects such as lakes, buildings or even trees as in the handball setting. For short, radio holes are always physical objects that interfere with the Wi-Fi mode such that some of the WiFi-connections are missing where they would usually exist without the presence of radio holes.

Of course, there are lots of routing algorithms that route data on competitive paths from a source to a target when there is no more knowledge than the geographical position of the source, the smartphone connected via Wi-Fi connections, and the target, see Chapter 3.

Greedy Routing. In general, local routing strategies such as greedy routing, where in every routing step you choose the smartphone that is closest to the target phone, work well to guarantee sending messages from source to target. In the context of the geometric routing of ad hoc networks, several routing techniques have been investigated.

These algorithms work well when there are no radio holes present. However, if there is a radio, sending the message that is always closer to the target means that, the message could get stuck at a radio hole and will possibly never arrive at the target. This is illustrated in the figure below.

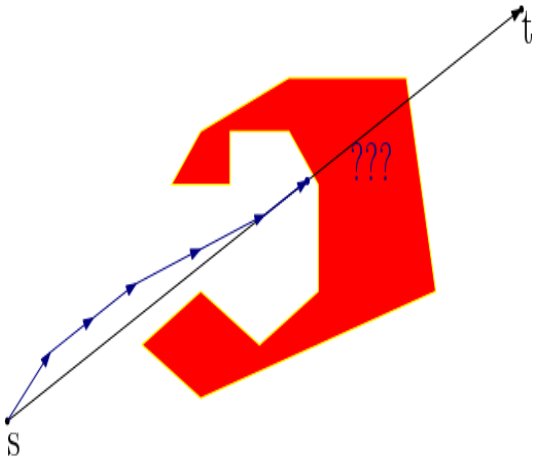

Figure 1.5: Greedy routing can lead to a dead end.

Here, the red shape represents the radio hole. When routing on the blue path from s to t, the last blue arrows stuck on the border of the radio hole. Even though the direction to t is clear, there is no way back to s via greedy routing to overcome the hole.

These routing algorithms find competitive paths only if all Wi-Fi connections are available. Thus, they are not applicable, when connections are missing, for example, due to radio holes. When data is routed from a source into the direction to the target and there is a radio hole between source and target, the data gets stuck at the hole and would never reach the target. In addition, routing around the obstacle on competitive paths around in advance is impossible, because of the restricted knowledge we have about the network, we do not know where to route around the obstacle. Our needs are clear, we need to gain much more information about the unknown network.

1.5 SetCover

In addition to designing an overlay that produces c-competitive routing paths, in this thesis, we use the hybrid communication model to design a distributed algorithm that solves SetCover. SetCover is a well-known problem in the centralized and distributed setting. Given are a collection of elements $\mathscr{U} := \{e_1, \ldots, e_n\}$ and sets $\mathscr{S} := \{s_1, \ldots, s_m\}$ with $s_i \subseteq \mathscr{U}$. The aim is to cover all elements with as few sets as possible. SetCover has many applications in the areas of computer science. On the one hand, it is important for the analysis of large data sets. The analysis of large data sets is needed in operations research, machine learning, information retrieval, and data mining, see [20]. On the other hand, the problem is well-known in distributed domains such as ad hoc sensor networks. An essential goal in these networks is to determine a minimum set of nodes. This set is called DominatingSet, i.e., all nodes are in a sensor range of this set. This set

fulfills different kind of tasks such as routing, collecting sensor data from neighbors, and others. DominatingSet is a special case of SetCover, i.e., all sets are also elements.

In the centralized setting, a greedy algorithm that chooses the set that covers most elements has only a logarithmic approximation factor. This is the best we can have for a polynomial-time algorithm unless $P = NP$. In the distributed setting, there are different randomized algorithms in the Congest model. The randomized algorithms match the optimal approximation ratio w.h.p.. These algorithms provide a near-optimal runtime of $O(\log(\Delta)^2)$. Here, Δ is the maximum degree of a set [22, 28]. For the setting of distributed algorithms, an instance of SetCover is often modeled as a *problem graph* $G_P := \{V_{\mathscr{U}} \cup V_{\mathscr{S}}, E\}$. This means that each set and each element corresponds to a node in G_P. For each set $s_i \in \mathscr{S}$ there exist a bidirected edge $\{e_j, S_i\} \in E$ to each element $e_j \in \mathscr{U}$ it contains. The edges represent a bidirected communication channel between the nodes that model the set and element in the default distributed setup. In each round of communication, each set and each element send a distinct message of size $O(\log(n))$ via each communication edge.

There are many results considering the runtime of algorithms in the CONGEST model. Nevertheless, there are only a few that deal with the message complexity. The message complexity is the number of messages that are needed to compute a distributed solution to SetCover. In all known solutions to SetCover, all sets and elements communicate extensively with their neighbors, i.e., they send messages to all of their neighboring sets and elements. This means a message complexity of $O(|E|)$ per round and $\tilde{O}(|E|)$. From the practical point of view, the high message complexity makes these algorithms less usable for dense networks in which sending only a single message is considered costly. One example are ad hoc networks. In ad hoc networks one needs to take care of the limited battery life of nodes. Thus, reducing sending messages to a minimum because of the high energy requirements of the radio module is of great interest. Hence, a distributed message-efficient algorithm for SetCover can solve this problem and reduce this energy consumption.

1.6 Research Questions

The huge advantage of using a communication mode for larger distances is to overcome not only large distances but also to find suitable paths around obstacles.

As mentioned above, by using hand signs, Jan can communicate to Philip or Alex to let them know that a path around the trees exists. Finally, using this mode guarantees finding a path around obstacles where the obstacles that possibly avoid playing while using only the communication mode on short distances.

The goal of this thesis is finding competitive paths with respect to the Euclidean distance in the ad hoc network while only knowing the geographic coordinate of the source and the target. To achieve knowledge about the ad hoc network, we will use the cellular infrastructure, the red connection between the masts, to exchange control information about the entire ad hoc network such as the location and shape of radio holes. These long-range connections available by the internet mode, will give us the direction of routing around radio holes. The routing itself takes place through Wi-Fi connections in the ad hoc network via the ad hoc mode.

Recall that the advantage of wireless ad hoc networks is that there is no limit, other than the bandwidth and battery constraints, on the amount of data that can be exchanged while the amount of data that can be transferred at a reasonable rate via long-range links using the cellular infrastructure or satellite is limited or costly. You as a smartphone owner can imagine these costs as the monthly fee you pay for a contract with a fixed internet volume. Of course, you do not wish to exceed this volume, otherwise you have to pay additional costs. Besides avoiding additional contract costs, another advantage of using internet-connections sparingly, is "green computing", i.e., using Wi-Fi instead of internet is more battery efficient. This means, we could save a huge amount of energy for charging the smartphone batteries, which would have a significant impact for the environment. Thus, the goal of this thesis addresses the following two research questions:

1. ***Can long-range links be used effectively to find competitive routing paths in the ad hoc network?***

and

2. ***Can we find a distributed* SetCover *algorithm that sends*** $o(|E|)$ ***messages?***

1.7 Thesis Overview

In this thesis, we present c-competitive routing protocols for some constant c from a source s and a target t (Section 1.1) in a network with holes (Section 1.3) in the hybrid communication model (Section 1.2). In addition, we present a distributed algorithm for SetCover (Section 1.5). Both topics can be read independently from each other.

Chapter 2 presents the models, the hybrid communication model (Model 2.1), and the main theorems Theorem 2.4, Theorem 2.5, Theorem 2.6, and Theorem 2.7). The model and theorems mentioned are published in

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Brief announcement: Competitive routing in hybrid communication networks. In: *Proceedings of the 30th on*

Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018, pages 231–233. ACM, 2018, cf. [23]

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Competitive routing in hybrid communication networks. In: *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments or Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers, volume 11410 of Lecture Notes in Computer Science, pages 15–31. Springer,* cf. [24]

2020 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020, pages 14:1–14:10. ACM,* cf. [6]

2019 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In: *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings, volume 11639 of Lecture Notes in Computer Science, pages 345–348. Springer,* cf. [5]

Further, this chapter presents the KT_0-Model (Model 2.1) to solve SetCover and the main Theorem 2.8 to solve this problem. Both the model and the main theorem, are published in

2021 (with T. Götte, C. Scheideler, and J. Werthmann) Beep-And-Sleep: Message and Energy Efficient Set Cover. In: *Algorithms for Sensor Systems - 17th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2021, Lisbon, Portugal, September 9-10, 2021, Proceedings* cf. [18]

Chapter 3 contains the related work about earlier routing approaches (Section 3.1) and Section 3.2. Further, it covers related work in hybrid communication models Section 3.3. This chapter rounds up with related work according to SetCover in Section 3.4.

Chapter 4 contains the preliminaries for this thesis and presents our first lemma. Section 4.1 surveys the state of the art on competitive routing in Delaunay triangulations, i.e., competitive routing without holes. In Section 4.2, we present preliminaries on polygonal routing. We also proof our first lemma in this thesis, which makes c-competitive routing in visibility graphs possible: see Lemma 4.8. This is published in

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Competitive routing in

hybrid communication networks. In: *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments or Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers, volume 11410 of Lecture Notes in Computer Science, pages 15–31. Springer,* cf. [24]

Section 4.3 presents preliminaries on a parallel convex hull algorithm. Finally, Section presents the Chernoff bounds which will be important later on for proofs of the algorithm for *SetCover*.

Chapter 5 presents the work on c-competitive routing with holes. Section presents the graph class that models the ad hoc network with holes and proves c-competitive path between any source s and target t of that network even in the presence of holes. Furthermore, this sections presents a routing protocol to find the latter mentioned paths. The results of this chapter are published in

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Brief announcement: Competitive routing in hybrid communication networks. In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018, pages 231–233. ACM, 2018,* cf. [23]

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Competitive routing in hybrid communication networks. In: *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments or Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, Au- gust 23-24, 2018, Revised Selected Papers, volume 11410 of Lecture Notes in Computer Science, pages 15–31. Springer,* cf. [24]

Chapter 6 presents the space reduction of convex hulls as holes abstractions instead of considering the whole hole in Section 6.1. It also presents proofs for the existence of c-competitive paths between a source s and a target t which is positioned on a convex hull or outside of it in Section 6.2. In addition, this chapter presents a c-competitive routing protocol for any source s and target t that are on convex hulls or outside of them in section. Furthermore, Section **??** presents the algorithm to obtain this kind of path. The results of this chapter are published in

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Brief announcement: Competitive routing in hybrid communication networks. In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018, pages 231–233. ACM, 2018,* cf. [23]

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Competitive routing in hybrid communication networks. In: *Algorithms for Sensor Systems - 14th In-*

ternational Symposium on Algorithms and Experiments or Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, Au- gust 23-24, 2018, Revised Selected Papers, volume 11410 of Lecture Notes in Computer Science, pages 15–31. Springer, cf. [24]

Chapter 7 presents bounding boxes as hole abstractions. Section 7.2 provides the proofs for the existence of c-competitive paths between any source s and target t that is outside of bounding boxes. Section 7.3 contains the routing protocol for the previous mentioned s and t. The results of this chapter are published in

2020 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In *ICDCN 2020: 21st Interna- tional Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020, pages 14:1–14:10. ACM,* cf. [6]

2019 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In: *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings, volume 11639 of Lecture Notes in Computer Science, pages 345–348. Springer,* cf. [5]

2017 (J. Sundermeier) Routing in Hybrid Communication Networks with Holes - Considering Bounding Boxes as Hole Abstractions. In: *Master Thesis, Universit at Paderborn*cf. [40]

Chapter 8 extents the research on bounding boxes. Section 8.1 provides the existence of a path through the intersection of bounding boxes between source s and target t that are outside of bounding boxes. Section 8.2 presents a routing protocol for the above mentioned path that is in the intersection of two bounding boxes. Section 8.3 gives an insight for the routing between s and t that are outside bounding boxes and the routing path goes through multiple intersecting bounding boxes. The results of this chapter are published in

2020 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In *ICDCN 2020: 21st Interna- tional Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020, pages 14:1–14:10. ACM,* cf. [6]

2019 (with J. Castenow and C. Scheideler) A bounding box overlay for competitive routing in hybrid communication networks. In: *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings, volume 11639 of Lecture Notes in Computer Science, pages 345–348. Springer,* cf. [5]

Chapter 9 gives an overview on how the knowledge of the unknown network is obtained. The results of this section are published in

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Brief announcement: Competitive routing in hybrid communication networks. In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018, pages 231–233. ACM, 2018,* cf. [23]

2018 (with D. Jung, C. Scheideler, and J. Sundermeier) Competitive routing in hybrid communication networks. In: *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments or Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, Au- gust 23-24, 2018, Revised Selected Papers, volume 11410 of Lecture Notes in Computer Science, pages 15–31. Springer,* cf. [24]

Chapter 10 presents a distributed algorithm for *SetCover*. The results are published in

2021 (with T. Götte, C. Scheideler, and J. Werthmann) Beep-and-sleep: Message and energy efficient set cover. ALGOSENSORS, 2021. [18]

Chapter 11 presents the conclusion of this thesis and provides an insight into future work related to this thesis.

2 Model and Main Results

In the previous chapter, we mentioned that online routing algorithms do not perform well, due to the missing knowledge about their shape. Sure, one could store all hole nodes and thus know more about the position and shape of these holes. This would require too much storage. For this reason, we are interested in hole abstractions that consist of fewer nodes. For hole abstractions we consider first convex hulls and then we use bounding boxes to reduce the number of nodes of convex hulls. These abstractions can reduce the storage significantly. In addition, we design a c-competitive routing algorithm for some constant c. This chapter highlights the model and main results for the this approach.

2.1 Model

Hybrid Communication Model for Competitive Routing.
Let $V \subset \mathbb{R}^2$ be a static set of nodes where the nodes have fixed positions in the Euclidean plane and $|V| = n$. Fixed positions V are reasonable, because the computation of the abstraction of the ad hoc network is so fast such that the geographic positions of the nodes do not change much in the meantime. For a given pair of nodes $u = (x_1, y_1)$, $v = (x_2, y_2)$, we call $||uv|| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ the Euclidean distance between u and v.

We model the smartphone ad hoc network as a hybrid directed graph $H = (V, E, E_{AH})$. The node set V models the set of smartphones. An edge (v, w) belongs to E whenever v knows the phone number or the *ID* of w. In addition, an edge $(v, w) \in E$ is in the *ad hoc edge set* E_{AH} whenever v can send a message to w via its Wi-fi. For all edges $(v, w) \in E \setminus E_{AH}$, v makes use of a long-range link to directly send a message to w. For simplicity, we use the *unit disk graph model* for the edges in E_{AH}.

Definition 2.1. For any point set $V \subseteq \mathbb{R}^2$ the *unit disk graph of V, UDG(V)*, is a bidirected graph that contains all edges (u, v) with $||uv|| \leq 1$.

The unit disk graph provides the advantage that one can simplify the definition of the hybrid graph to $H = (V, E)$, because the ad hoc edges are determined by the positions of V. Assume UDG(V) is strongly connected. This means that every node of V can send a message to any other node in V using ad hoc edges. The ad hoc edges are fixed but the nodes can change E over time, for example, if a node

v knows the IDs of the nodes w and w', as they are neighbors in H, it can send the ID of w to w'. This adds (w, w') to E. Also, if v deletes the address of node w with $(v, w) \in E$. This means that (v, w) is removed from E. We do not consider any other kind of changes in E, for example, like changes such that a node v learns about an ID of a node w unless w is in v's UDG-neighborhood or changes such as the ID of w is sent to v by some other node.

We consider *synchronous message passing* in which time is divided up into rounds. More precisely, we consider that every message initiated in round i is also delivered at the beginning of round $i + 1$. We assume that a node can process all messages in a round that have been delivered at the beginning of that round.

KT_0-Model for SetCover. This model and also the problem statement is independent from the models and problems mentioned above. We use this KT_0 model to solve SetCover. In this model we have a fixed communication graph $G := (V_S \cup V_{\mathscr{U}}, E)$ with $V_S = n$ and $V_{\mathscr{U}} = m$ with bidirected communication edges $\{s, e\} \in E$ between a set and its elements. Sets and elements can locally distinguish between their edges through port numbers, but there are no global identifiers that uniquely identify a node. For simplicity, all nodes know Δ and $\log(n + m)$. Also, here the time proceeds in synchronous *rounds*. In each round, each element or set can send a distinct message of size $O(\log(n))$ to any subset of its neighbors. The messages are received in the next round. In addition, all nodes wake up in the same round.
In related literature, this model is called the *clean network model* [36]. Note that this model does not intend to represent ad hoc networks faithfully. Instead it is used to analyze the message efficiency of distributed algorithms.

2.2 Main Results

We consider a hybrid graph $G = (V, E, E_{AH})$ where the unit disk graph of V is connected.

The following figure provides an overview of the types of and reasons for the various kinds of hole abstractions.

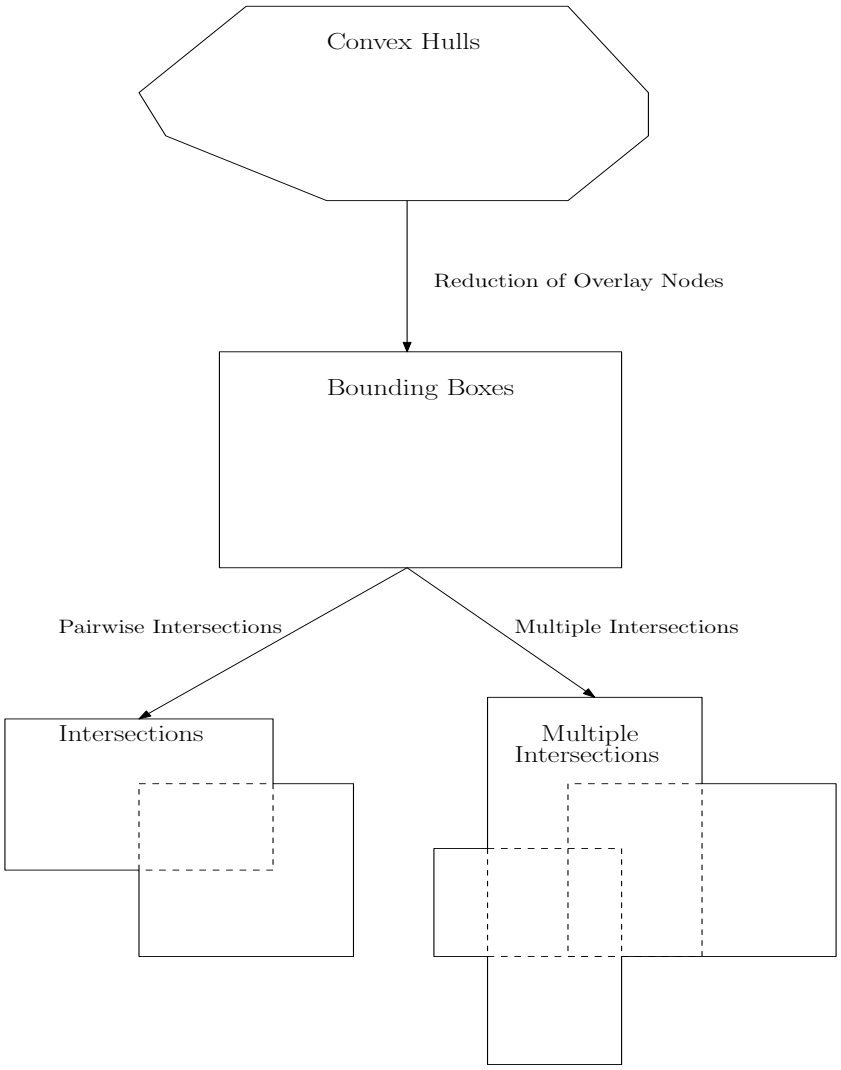

Figure 2.1: Main results for routing algorithms.

The first step is to compute convex hulls as hole abstractions. A convex hull is defined as follows [37]

Definition 2.2. A set of points in a Euclidean space $\mathbb{R}^2$ is defined to be convex if it contains the line segments connecting each pair of its points. The convex hull of

a given set X is defined as the (unique) minimal convex set containing X.

One can imagine stretching a rubber band so that it surrounds the entire set X and then releasing it, allowing it to contract. When it becomes tight, it encloses the convex hull of X.

In the second step, our aim is to reduce the number of convex hull nodes, which we achieve by calculating bounding boxes as hole abstractions.

The following definition provides the axis-parallel bounding box of a hole.

Definition 2.3 (Bounding Box)**.** Let p be a polygon. Let $min_x = \min_{v \in p}(x(v))$ and max_x, min_y and max_y be defined analogously. These points are called *extreme points* of p. The (axis-parallel) *Bounding Box* of p is a polygon with the following nodes:

1. $bb_{t\ell}(p) = (x(min_x), y(max_y))$ (top-left)
2. $bb_{tr}(p) = (x(max_x), y(max_y))$ (top-right)
3. $bb_{b\ell}(p) = (x(min_x), y(min_y))$ (bottom-left)
4. $bb_{br}(p) = (x(max_x), y(min_y))$ (bottom-right)

The nodes are connected via the direct line segments $\overline{bb_{t\ell}(p)bb_{tr}(p)}$, $\overline{bb_{tr}(p), bb_{br}(p)}$, $\overline{bb_{t\ell}(p)bb_{b\ell}(p)}$ and $\overline{bb_{b\ell}(p)bb_{br}(p)}$.

After we reduced this number of nodes, the third step is to investigate also routing algorithms between any pair of nodes that is outside of the hole abstraction and whose shortest path goes through intersections of hole abstractions. Therefore, we present a routing algorithm that finds competitive paths through pairwise intersections of bounding boxes. Afterwards, in step 4, we consider the topic of routing through multiple intersections of bounding boxes.

More concretely, in the setting for convex hulls as radio hole abstraction, we consider H to be the set of radio holes in G, and C to be the set of convex hulls of radio holes in H. The length of the perimeter of a radio hole $h \in H$ is denoted by $P(h)$ and the circumference of a minimum bounding box of a convex hull $c \in C$ is denoted by $L(c)$. Further, we call the nodes and edges of G with geographic positions between a convex hull edge and the chain of nodes between two adjacent convex hull nodes BAY AREA.

Our main result for convex hulls as radio hole abstraction is:

Theorem 2.4. *Assume any distribution of the nodes in V that ensures that the UDG(V) is connected and of bounded degree. Further assume that the convex hulls of the radio holes do not overlap.*

Then our algorithm computes an abstraction of UDG(V) in $\mathcal{O}(\log^2 n)$ communication rounds using only polylogarithmic communication work at each node so that c-competitive paths between all source-destination pairs whose geographical position is outside of convex hulls can be found in an online fashion.

The space needed by the convex hull nodes of the radio holes is $\mathcal{O}\left(\sum_{c\in C} L(c)\right)$. Nodes lying on the boundary of radio holes need storage of size $\mathcal{O}\left(\max_{h\in H} P(h)\right)$. For every other node, the space requirement is constant.

Furthermore, we reduce the number of nodes of hole abstractions to a constant, by presenting bounding boxes as radio hole abstraction.

The main result for bounding boxes as radio hole abstraction is the following

Theorem 2.5. *For any distribution of the nodes in V that ensures that UDG(V) is connected and of bounded degree, where the bounding boxes of the radio holes do not overlap, our algorithm computes an abstraction of UDG(V) in $\mathcal{O}(\log^2 n)$ communication rounds using polylogarithmic communication rounds at each node such that we obtain 18.55-competitive paths between all source-destination pairs whose geographical position is outside of bounding boxes.*

For routing through intersections of hole abstractions, we present the following for pairwise intersecting bounding boxes

Theorem 2.6. *For any distribution of the nodes in V that ensures that UDG(V) is connected and of bounded degree, where the bounding boxes of the radio holes do overlap pairwise and the convex hulls of holes do not overlap, our algorithm computes an abstraction of UDG(V) in $\mathcal{O}(\log^2 n)$ communication rounds using only polylogarithmic communication rounds at each node to obtain 28.83-competitive paths between all source-destination pairs with geographical position outside of bounding boxes.*

Both for intersecting and non-intersecting bounding boxes, we provide a heuristic solution and show via simulations that our approach significantly outperforms classical online routing strategies for ad hoc networks with holes.

Theorem 2.7. *In simulations, for any distribution of the nodes in V that ensures that UDG(V) is connected and of bounded degree, where the bounding boxes of the radio holes overlap and the convex hulls of holes do not overlap, our algorithm computes an abstraction of UDG(V) in $\mathcal{O}(\log^2 n)$ communication rounds using only*

polylogarithmic communication rounds at each node to obtain $(10.68 + c \cdot 18.55)$*-competitive paths between all source-destination pairs whose geographical position is outside of bounding boxes.*

Further, we present a distributed algorithm that solves SetCover and prove the following

Theorem 2.8. *There is an algorithm in the* KT_0-Congest *model that solves* SetCover *in time* $O(\log^2(\Delta))$*, the expected approximation ratio of* $O(\log(\Delta))$*, and sends only* $\tilde{O}(\sqrt{\Delta}(n+m))$ *messages, w.h.p., given that all nodes know* Δ *and an approximation of* $\log(n)$*.*

We also prove a lower bound

Lemma 2.9. Any algorithm that yields an $O(1)$-approximation for SetCover needs at least $\tilde{\Omega}(m\sqrt{n})$ messages on certain graphs in the KT_0 model.

3 Related Work

3.1 Geometric/ Local Routing

In the previous chapter, we already mentioned Greedy routing and showed an example that leads to a dead end. There is, of course, related work on algorithms that improved Greedy routing.

Compass Routing. Compass Routing [26] is a similar approach. The algorithm considers the direct line segment which connects the source node s and the target node t. At every routing step the edge with smallest slope to the direct line segment is picked. This, however, does not lead to a message delivery guarantee in all the different kinds of graphs.

Face Routing. There are routing approaches to overcome obstacles for a message delivery guarantee to the target. These routing approaches are based on FACE ROUTING. FACE ROUTING explores the boundary of faces in planar graphs by the right-hand rule which means that it explores this boundary in counterclockwise direction. An example for this is following the right hand wall in a maze. This algorithm explores the whole boundary of faces.

GPSR. A routing algorithm that combines FACE ROUTING and GREEDY ROUTING is called GPSR [25]. The basic idea is, whenever the message arrives at a hole, the message is directed via FACE ROUTING around the face of the hole until greedy routing is possible again from the obstacle to the target (see the following figure).

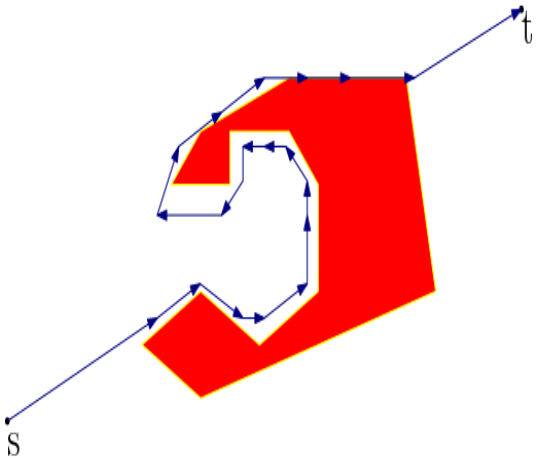

Figure 3.1: First greedy routing and then face routing can lead to long detours.

As soon as greedy routing is possible again, the routing mode changes to greedy routing.

Compass Routing II. The authors in [26] introduce a strategy called COMPASS ROUTING II. This strategy combines compass routing with face routing with delivery guarantee for all different kinds of connected geometric graphs.

Extensions of GPSR. There are a couple of extensions of these original routing ideas of GPSR and COMPASS ROUTING II that combine common routing algorithms with face routing. One of these extensions is AFR (adaptive face routing). It was published in [29] and is based on a bounded face routing algorithm BFR to avoid routing along the whole boundary of a face. To do so an ellipse is chosen such that its size contains the complete optimal path. After hitting the ellipse, the algorithm turns back to the face until it hits the ellipse again. Further, the second hit completes the face exploration. One problem of applying only BFR in unknown graphs is that the optimal path is not known in advance. A solution for this problem is that AFR starts with BFR with an initial ellipse size of the estimated optimal path between source s and target t. In case the algorithm does not reach t, it starts again with the double size of the ellipse. The authors proved that AFR is asymptotically optimal in unit disk graphs. Other extensions are OAFR (other adaptive face routing), GOAFR (greedy other adaptive face routing) [31], and GOAFR+ [4] [30]. In [30] and [29,31], the authors proved that GOAFR and GOAFR+ are asymptotically optimal. The algorithms GOAFR and GOAFR+ achieve a path length with a quadratic competitiveness compared to the shortest path. The latter three mentioned algorithms will play a role in this thesis. We will compare a simulation with those algorithms.

A different approach from local routing is flooding and can be discribed as follows:

Flooding. To overcome problems which avoid a guaranteed delivery, a restricted flooding procedure is chosen. In the setting of nodes on the grid, a packet is routed along multiple paths and is thus comparable to a restricted flooding procedure [38]. In their model, the authors consider the existence of alive nodes and crashed nodes on the grid. The crashed nodes can be compared with the obstacles on the grid which should be avoided by routing paths.

If there are no holes present, there are routing algorithms that route on c-competitive paths in certain triangulations.
MixedChordArc. A graph with message delivery guarantee is the Delaunay graph, which in addition, is a 1.998-spanner of the Euclidean metric [41]. Fundamental research on competitive routing in Delaunay triangulations is published by Bose et al. [3]. The authors present a 3.56-competitive routing algorithm MixedChordArc for a certain type of triangulations. This algorithm will play an important role in this thesis. We will use it whenever routing in areas without radio is possible. For more details see section 4.1.

3.2 Global Knowledge

INF. INF [9] is a routing approach that combines greedy forwarding with randomness. If a packet gets stuck via greedy routing, then a random intermediate location is chosen. This approach requires some use of a global knowledge to choose a random node that is not too far away from the location. In addition, INF does not ensure a delivery guarantee. To solve this problem all nodes regularly post information about their geographic position and the nodes within their communication range to inform a server on the Internet. This allows the server to compute optimal routing paths which means that whenever a node wants to forward a message to a certain destination, the server can inform it which of the neighbors to send it to.

BoundHole. There is also BoundHole [13] as an example of routing strategies that use a portion of global knowledge about the network. BoundHole uses a preprocessing phase at each node that is located at the boundary of a hole. These hole nodes send a packet and this packet is routed using the right-hand rule around the hole until it reaches the source of the message. On its way, the packet collects information about the hole boundary. With the knowledge about the hole boundary, the authors find shorter paths than strategies which only use local information.
For a survey on all mentioned strategies, we refer to [1].

3.3 Hybrid Communication Networks

To combine local and global routing strategies, where the goal is to use only little global knowledge, we will introduce the Hybrid Communication Network. Hybrid

Communication Networks have been considered in different contexts. In practical applications, the term Hybrid Communication Network usually combines wired with wireless networks, see [7]. Close to the application in this thesis is the scenario presented in [16]. The authors assume an external network which is not controlled by the network participants. But the participants can control an internal network. The authors show that the combination of the two networks allows evaluating monitoring problems of the external network faster than in the approaches which use the links of the external network.

Using local edges, Augustine et al. [2] model a fixed communication network in which $\mathcal{O}(1)$ messages of size $\mathcal{O}(\log n)$ can be sent over every edge in each synchronous round. The global edges form a clique. Nodes are only allowed to send and receive a total of at most O(log n) messages over global edges. The authors investigate the problem of computing shortest paths in the graph given using the local connections. For the all-pairs shortest paths problem, they obtain that an exact solution can be computed in time $\tilde{O}(n^{\frac{2}{3}})$ and that the minimum time to compute approximate solutions is $\tilde{O}(\sqrt{n})$. They prove that for the single-source shortest paths problem a solution can be computed in time $\tilde{O}(\sqrt{SPD})$, where SPD denotes the shortest path diameter.

The authors of [14] present algorithms in the hybrid communication model to compute single-source shortest paths and the diameter that are very efficiently in sparse graphs. They present $\mathcal{O}(\log n)$ time algorithms for cactus graphs. Cactus graphs are graphs in which each edge is contained in at most one cycle, and 3-approximations for graphs that have at most $n + \mathcal{O}(n^{1/3})$ edges and arboricity $\mathcal{O}(\log n)$. For these graph classes, their algorithms calculate exponentially faster solutions than the best known algorithms for general graphs.

3.4 SetCover

There is only some related work on the message complexity of SetCover.

SetCover in streaming models. The space complexity of SetCover in streaming models is a topic with results for different variants of the problem. In this model, the edges of the problem graph and thus the information on which element belongs to which set, arrive sequentially. An algorithm stores each edge for later and iterate over the whole input several more times. The aim is to solve SetCover using only as few passes and as few spaces as possible. The difference is that a trivial algorithm can just iterate over the input once and stores all edges, and then solve the problem using an *offline* algorithm.

SetCover solved with randomized algorithm. Demaine et al. further showed that randomization is of interest for a space complexity of $o(mn)$ as it is impossible for a deterministic algorithm.

SetCover in a distributed model. The authors Indyk et al. used streaming algorithms in the area of sublinear algorithms [20]. Their goal is to find an algorithm that makes as *few* queries as possible and provides a good approximate solution. This model has something in common with our distributed model. In their case, the number of queries is $o(mn)$, i.e., sublinear in the number of edges. Indyk et al. present a polynomial-time algorithm that makes $\tilde{O}(m+\sqrt{m}n)$ queries and gives an $O(\log^2(n))$-approximate solution. With exponential runtime, the approximation ratio is even reduced to $O(\log^2(n))$, but the algorithm still needs $\tilde{O}(m+\sqrt{m}n)$ queries. Furthermore, Indyk et al. gave also evidence that $\Omega(nm^\epsilon)$ queries are necessary to achieve good approximation ratios.

SetCover in further distributed models. Further fully distributed approaches for SetCover and related optimization problems were published by Jia et al. [22] as well as Kuhn and Wattenhofer [28]. They presented fast distributed algorithms to solve the DominatingSet problem in time in $O(\log(n)^2)$. Their goal is to find the minimal set of nodes adjacent to all nodes in the graph $G := (V, E)$. The DominatingSet problem is a special case of SetCover where each node is both an element and a set. Kuhn et al. presented also an algorithm to solve SetCover in $O(\log(n)^2)$ rounds in the CONGEST model [27]. They prove that this is close to the optimal runtime of $O(\log(n))$.

Roughgarden et al. presented a distributed algorithm for convex optimization problems. This includes SetCover and it works on any eventually connected communication network [35]. In particular, it works on dynamic communication networks. That means that they can change from round to round. This algorithm is fully distributed but it heavily relies on sequential calculations to cope with these general communication networks. Thus, its runtime and message complexity are polynomial in the number of nodes.

DominatingSets in ad hoc networks. There is quite a lot of work done on the DominatingSet-problem according to ad hoc networks. The work by Scheideler et al. [39] considers the SINR model where the nodes are modeled as points in the Euclidean plane. In addition, there is also a solution to the DominatingSet-problem in the Beeping-model for unit-disk graphs [43]. This algorithm has some similarities with ours but is not applicable to general graphs. Last but not least, the authors in [19] present a distributed algorithm in the beep model.

4 Preliminaries

Before we start investigating competitive routing with holes, we first introduce the preliminary concept of competitive routing without holes, namely in complete triangulations. Therefore, we introduce the Delaunay triangulation and the MixedChordArc algorithm. The MixedChordArc algorithm finds competitive paths in complete Delaunay triangulations. These are Delaunay triangulations without holes. We will adapt both later to the hybrid communication model for areas without holes in the ad hoc network.

In addition to competitive routing in complete Delaunay triangulations, we will introduce the preliminaries of polygonal routing because we will later consider convex hulls and bounding boxes as hole abstractions. You can imagine both as polygons.

Further, we mention the Chernoff Bound and provide more background information which will be used later for some proofs of SetCover. The figures of the Delaunay triangulation were designed by Alina Ergova.

4.1 Competitive Routing in Complete Triangulations

This section presents the preliminaries for competitive routing in hybrid communication networks without holes. We will consider the 2-localized Delaunay graph in the latter part of this thesis, i.e., the 2-localized Delaunay graph will represent the ad hoc network with holes. It is related to the Delaunay triangulation that we present in this section to obtain a better intuition about the 2-localized Delaunay graph. We also explain the MixedChord algorithm, which is known to be a 3.56-competitve algorithm for any two nodes of the Delaunay triangulation. *MixedChordArc* is fundamental for this thesis, because we will use this routing algorithm to route between certain nodes of the ad hoc network, where geometric routing is possible.

Throughout this thesis, we assume the set of nodes V to be non-pathological, i.e., there are no three nodes on a line and no four nodes on a cycle. We also assume that the coordinates of each node are unique and that there are no two nodes on the same geographic position. The Delaunay triangulation is defined as follows.

Definition 4.1. Let $\bigcirc(u, v, w)$ be the unique circle through the nodes u, v, and w and $\triangle(u, v, w)$ be the triangle formed by the nodes u, v, and w. For any $V \subseteq \mathbb{R}^2$, the

Delaunay graph $\mathrm{Del}(V)$ of V contains all triangles $\triangle(u, v, w)$ for which $\bigcirc(u, v, w)$ does not contain any further node besides u, v and w.

The following figure presents a complete Delaunay triangualtion.

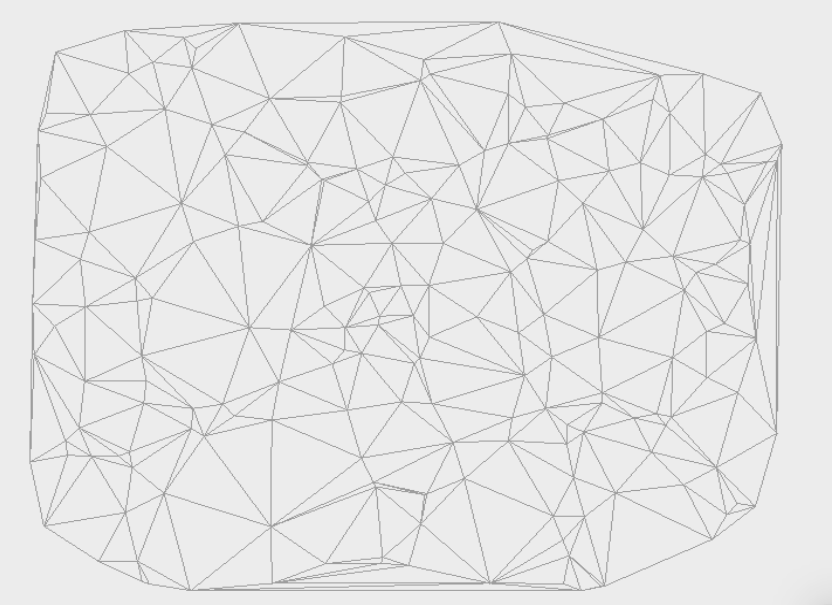

Figure 4.1: Delaunay triangulation.

The Delaunay graph $\mathrm{Del}(V)$ contains paths between every pair of nodes v and w of V which are not longer than c times their Euclidean distance. We call these paths as follows:

Definition 4.2. A *path* $(v, \ldots, w)$ between two nodes v and w in a geometric graph G is a *geometric c-spanning path* between v and w, if its length is at most c times the Euclidean distance between w and v.

Classes of graphs that contain such paths for every pair of nodes are called geometric c-spanners.

Definition 4.3. A graph $G = (V, E)$ is called a *geometric c-spanner*, if for all $v, w \in V$ there is a geometric c-spanning path $(v, \ldots, w)$ in G.

Delaunay graphs are known to be geometric c-spanners. The currently best known bound on c is 1.998 and was proven by Xia [41].

Theorem 4.4. *There exists a path in a Delaunay graph from node s to t of length less than* $1.998 \cdot \|st\|$.

As mentioned above, in this thesis, the online strategy *MixedChordArc* [41] is fundamental for the areas of the ad hoc network without holes. Bose et al. introduced the online routing strategy *MixedChordArc* for Delaunay Graphs which only considers edges of triangles that are intersected by the direct line segment between source and destination. The pseudocode for the *MixedChordArc* algorithm is as follows and it makes use of the notation, where C is a circle that intersects st and t_C is the rightmost point of C on st. Further u and v are two points on C and $\mathscr{A}_C(u,v)$ is the clockwise arc of C from u to v, and by $\mathscr{B}_C(u,v)$ is the counter-clockwise arc of C from u to v:

Algorithm 1 MixedChordArc

if $p = s$ **then** if $|\mathscr{A}_C(s,t_C)| \leq |\mathscr{B}_C(s,t_C)|$
$p \leftarrow a$,
otherwise $p \leftarrow b$
if $p \neq s$ repeat the following until $p = t$
1. if p is above $\overline{st}$ then

if $|\mathscr{A}_C(s,t_C)| \leq |pb| + |\mathscr{B}_C(s,t_C)|$ **then** $p \leftarrow a$.
Else $p \leftarrow b$.
2. if p is below $\overline{st}$ then

if $|\mathscr{B}_C(s,t_C)| \leq |pa| + |\mathscr{A}_C(s,t_C)|$ **then** $p \leftarrow b$.
Else $p \leftarrow a$.

Output: $\mathscr{P}_{MixedCordArc}\langle s,t\rangle$

In their work, the authors prove the following

Theorem 4.5. *The* MixedChordArc *algorithm finds a path in* $Del(V)$ *between any source* s *and target* t *with length at most* $3.56 \cdot \|st\|$.

The following figure shows a path that was chosen by MixedChordArc.

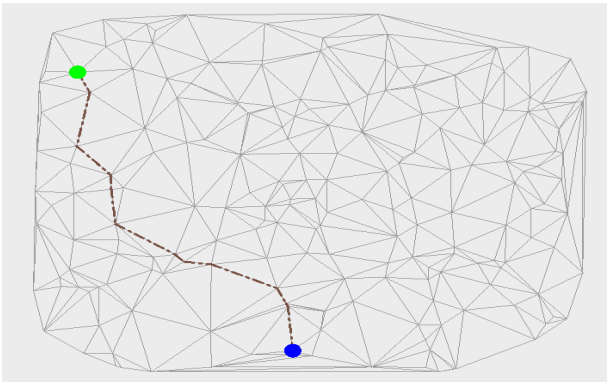

Figure 4.2: MixedChordArc.

4.2 Polygonal Routing

To find constant-competitive paths between any pair of nodes in the 2-localized Delaunay graph, we consider results from computational geometry.

We abstract holes of the 2-localized Delaunay graph with convex hulls and bounding boxes and this scenario is comparable to polygonal routing.

Polygonal routing considers a starting point s and a target point t in the Euclidean plane. The goal is to find a path in the plane between s and t. The challenge is the presence of polygonal obstacles that avoid walking directly along the line segment $\overline{st}$. In this thesis, polygonal obstacles represent radio holes.

De Berg et al. [11] considered nodes of obstacle polygons to find shortest paths in polygonals and proved

Lemma 4.6. Any shortest path between s and t among a set S of disjoint polygonal obstacles is a polygonal path whose inner nodes are nodes of S.

A common procedure for finding shortest paths between nodes of polygons is the computation of a visibility graph and then applying a single-source shortest path algorithm, for example with the algorithm of Dijkstra.

The visibility graph is defined as follows

Definition 4.7. In the visibility graph $Vis\,(V)$ of a set of polygons, V represents the set of corners of the polygon, and there is an edge $\{v, w\}$ in $Vis\,(V)$ if and only if a line can be drawn from v to w without crossing any polygon, i.e., v is visible from w.

The combination of the theorem and lemma mentioned above implies that

Lemma 4.8. The shortest path in the visibility graph of hole nodes of the 2-localized Delaunay graph (which would be the shortest possible geometric connection between the source and the target node) yields to a 3.56-competitive path in the 2-localized Delaunay graph by applying the *MixedChordArc* between every pair of consecutive nodes on the path.

4.3 Parallel Convex Hull Algorithm

This section refers to the parallel convex hull algorithm by Miller [34]. We use the parallel convex hull algorithm to compute the convex hull of the 2-localized Delaunay graph.

The hypercube is the requirement to apply the parallel algorithm by Miller [34] to calculate convex hulls and is defined as follows

Definition 4.9. A d-dimensional *hypercube* consists of n nodes, where $n = 2^d$, such that each node has a unique bitstring $(x_1, \ldots, x_d) \in \{0,1\}^d$ and there is an edge between two nodes if and only if their bitstring differs in only one bit. The decimal representation of a bitstring of a node h is denoted as $id(v)$.

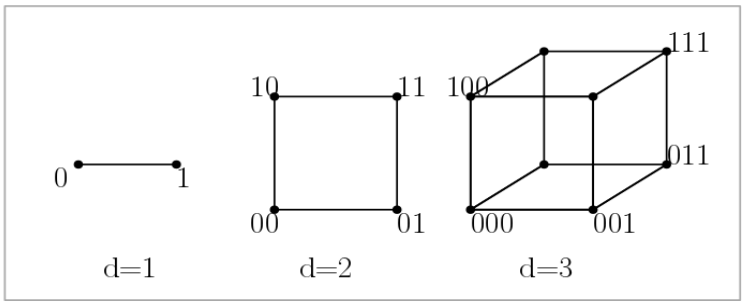

Figure 4.3: The hypercube for dimension 1, 2, and 3.

The advantage of this hypercube data structure is that the diameter as well as the degree of a hypercube is d and $d = \log n$ since $\log n$ bits are needed to represent n nodes.

Besides the hypercube, the protocol requires n sorted points corresponding to their x-coordinate. It is well-known that

Lemma 4.10. Identifying the convex hull of a planar set of points can not be done faster than sorting.

Sorting n points in a hypercube can be done in $\mathcal{O}(\log n)$ communication rounds on expectation with the algorithm of Reif and Valiant [34]. Upon termination, Miller's algorithm is applied which ensures that each node of the ring knows every convex hull node and especially each convex hull node identifies itself as a convex hull node. The following theorem follows:

Theorem 4.11. *Given a hole ring with k nodes, the convex hull of this hole ring can be calculated in $\mathcal{O}(\log k)$ communication rounds on expectation.*

If we use a deterministic sorting algorithm to sort the nodes according to their x-coordinate, we achieve

Theorem 4.12. *Given a hole ring with k nodes, the convex hull of this hole ring can be calculated in $\mathcal{O}(\log^2 k)$ communication rounds.*

To achieve the theorem above, the authors present a divide and conquer algorithm that in each round combines two convex sets S_1 and S_2. This algorithm uses the following lemma, published by [34]:

Lemma 4.13. Let $\overline{p_i q_j}$, where $p_i \in S_1$ and $q_j \in S_2$ are the upper tangent line between S_1 and S_2. Then the slope of $p_i q_j$ is between the slope of $\overline{p_{i-1} p_i}$ and the slope of $\overline{p_i p_{i+1}}$, and the slope of $\overline{q_{j-1} q_j}$ and the slope of $\overline{q_j q_{j+1}}$ respectively.

The following algorithm provides the high-level idea of the convex hull algorithm by Miller at al.

Algorithm 2 Parallel Convex Hull Algorithm

1. The set S of all planar points is divided into $n^{\frac{1}{2}}$ subsets $S_1, S_2, ..., S_{n^{\frac{1}{2}}}$, each of equal size, such that the x-coordinates of all points in S_i are less than the x-coordinates of all points in S_j, for $i < j$.

2. Let the regions R_i be the processors of points in S_i. For each R_i, $1 \leq i \leq n^{\frac{1}{2}}$, recursively identify the extreme points of the convex hull of S_i.

3. In $\Theta(\log n)$ time, each region R_i can determine the westmost extreme point p_{w_i} and the eastmost extreme point p_{e_i} of S_i.

4. The following steps are to determine for each region R_i the upper tangent line, and when performed a second time, the lower tangent line, between the set of points S_i and every other set of points S_j, for $i \neq j$:
   a) For each region R_i mark $n^{\frac{1}{4}}$ extreme points that are equidistantly placed with respect to the counterclockwise order of the extreme points of S_i. For each marked extreme point p_k, with respect to the counterclockwise ordering of extreme points in R_i, create two slope records $\overline{p_{k+1}, p_k}$ and $\overline{p_k, p_{k-1}}$.
   b) Each region R_i sends its slope records to every other region R_j.
   c) Every region R_i creates two slope records for each of its $\mathcal{O}(n^{\frac{3}{4}})$ extreme points.
   d) Now every Region record determines the largest slope of a hull edges that is smaller than its slope and the smallest slope of a hull edge that is larger than its slope
   e) The records are sent back to their origin regions ordered by destination region by slope.
   f) Steps 4a)-4e) are performed twice more.

5. In this step, every region R_i decides which and how many of its extreme points are extreme points of S:
   a) Determine the minimum slope of a tangent line between R_i and R_j.
   b) Determine the maximum slope of a tangent line between R_i and R_j.
   c) Let p_r be the extreme point of R_i. If p_r is to the left of p_l, or $p_r = p_l$ and the angle open to the top, formed by these two line segments, is less than 180°, then no points of R_i are extreme points of S. Otherwise, those extreme points of R_i between p_l and p_r are extreme points of S.

6. Every region R_i creates a record with the total number of points that remain in the final upper (lower) hull.

4.4 Chernoff Bounds

In this section, we mention the Chernoff Bounds. These inequalities will be important to prove lemmas for the design of the algorithm for SetCover.

Lemma 4.14 ((Chernoff Bounds.)). Let $X_1, ..., X_n$ be a set of independent binary random variables. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then it holds:

1. For any $\delta \geq 1$ it holds $Pr[X > (1+\delta)\mu] \leq exp\Big(\frac{-\delta\mu}{3}\Big)$.
2. For any $0 \leq \delta \leq 1$ it holds $Pr[X > (1+\delta)\mu] \leq \exp\Big(-\delta^2\frac{\mu}{3}\Big)$.
3. For any $0 \leq \delta \leq 1$ it holds $Pr[X \leq (1-\delta)\mu] \leq \exp\Big(-\delta^2\frac{\mu}{2}\Big)$.

Also the following lemma will be useful for that section. The main message of the following lemma is that for any node $u \in \mathscr{U}$ it holds that the probability that u is covered by more than t sets simultaneously exponentially declines in t. This is independent of the phase and round in which u is covered. For the ease of better readability, we refer to the number of sets that cover u simply as μ (and not as $\mu_i(u)$).

Lemma 4.15. For any value $t > 1$, any element $u \in V$, and any round in which u is covered, it holds:

$$\mathbf{Pr}\left[\mu(u) = t\right] \leq \max\{e^{-t/3}, 2\left(1 - 1\sqrt[k]{\Delta}\right)^{t-1}\} \tag{4.1}$$

4.5 SetCover

In this section, we describe the Beep-And-Sleep algorithm. The algorithm is called like this, because most sets and elements will be idle during the execution. The results are published in [18] and they are the preliminaries for the SetCover algorithm that we will present in chapter 10.

The authors of [18] use the following variant of the Beeping model [8, 12]:

1. They consider a fixed communication graph $G := (V_S \cup V_{\mathscr{U}}, E)$ with $V_S = m$ and $V_{\mathscr{U}} = n$. Each set $s \in V_S$ has a bidirected edge $\{s, e\} \in E$ to each element $e \in V_{\mathscr{U}}$ it contains. Each node can only communicate with its neighbors in G. Further, all nodes know Δ, the maximum degree of G. This assumption can be replaced by a polynomial upper bound, i.e., an approximation of $\log \Delta$, which would only slow the algorithm down by a constant factor. Note that the nodes do not know their exact degree, and nodes have *no* identifiers.
2. Time proceeds in so-called *slots*. In each slot, a node can either *beep* or *listen*. If a node *listens* and any subsets of its neighbors *beeps*, the *listening* node

Algorithm 3 BEEP-AND-SLEEP: CODE FOR SETS

procedure SETS(s, N_s, k, Δ)
  $S_s \longleftarrow 0$
  $A \longleftarrow \{\perp\}^{4k}$
  for $i := 0, \ldots, k$ **do**
    Pick $X_s \sim Geo(1 - \frac{1}{\Delta^{\frac{1}{k}}})$.
    $X_s \longleftarrow \min\{X_s, 4k\}$

    for $j := 0, \ldots, 4k$ **do**
      for $\ell = 1, \ldots, 4k$ **do**

        $A_\ell \longleftarrow$ LISTEN()

      end for
      if $(|\{\ell \mid A_\ell \neq \perp\}| \geq 3k) \wedge (4k - X_s = j)$ **then**
        Send BEEP to all neighboring elements.
        $S_s \longleftarrow 1$ ▷ Add s to S.
      end if
      $A \longleftarrow \{\perp\}^{4k}$
    end for
  end for
end procedure=0

Algorithm 4 BEEP-AND-SLEEP: CODE FOR ELEMENTS

procedure ELEMENTS(e, N_e, k, Δ)
  $C_e \longleftarrow 0$

  for $i := 0, \ldots, k$ **do**
    for $\ell = 1, \ldots, 4k$ **do**
      $Y_e^\ell \sim B\left(\frac{1}{\Delta_i}\right)$ ▷ $\Delta_i := \Delta^{\frac{k-i}{k}}$
    end for
    for $j := 0, \ldots, 4k$ **do**
      for $\ell = 1, \ldots, 4k$ **do**
        if $(Y_e^\ell = 1) \wedge (C_e = 0)$ **then**
          Send BEEP to all neighboring sets.
        end if
      end for

      $l \longleftarrow$ LISTEN()
      If $l \neq \perp$, set $C_e \longleftarrow 1$

    end for
  end for
end procedure

Figure 4.4: The pseudocode depicts the code for the elements (right) and the code for the sets (left). Sets and elements are synchronized. Whenever the sets beep, the elements listen, and vice versa.

receives a BEEP. It can neither distinguish which neighbors beeped nor how many neighbors beeped, i.e., it only relies on carrier sensing. Further, a node *cannot* simultaneously beep and listen but must choose one of the two options.

3. All nodes wake up in the same slot, i.e., we consider the BEEPING model with simultaneous wake-up. We believe that our algorithm can also be extended for wake-up-on-beep as each node only needs to be in sync with neighboring nodes. If nodes do not wake up in the same round, their internal counters only differ by 1 as each node wakes up one slot after its earliest neighbor. In this case, there are some standard tricks to simulate a single slot of a simultaneous wake-up algorithm within 3 slots [8, 12].

With this model, the authors show the following:

Theorem 4.16. *There is an algorithm in the* BEEPING *model that solves* SET-COVER *in time* $O(k^3)$ *with approximation ratio* $O(\Delta^{\frac{3}{k}} \log^2 \Delta)$ *where* $k > 3$ *is a parameter known to all nodes.*

Hence, even for a constant k, the algorithm achieves a non-trivial approximation ratio. This result is close to the optimal ratio with this runtime, because Kuhn et al. proved that any distributed algorithm with local communication needs $O(k)$ time for an approximation ratio of $O(\Delta^{\frac{1}{k}})$ [27].

The main idea of this algorithm is that all sets that cover the *most* uncovered neighbors at a given point in time add themselves to the solution. Because of the constraints of the BEEPING model, the uncovered elements cannot tell their neighboring sets that they are uncovered. If all elements were able to send a BEEP simultaneously, the listening sets would only learn that they have at least one uncovered neighbor but they would not learn how many uncovered neighbors they have. Since the algorithm works in a distributed setting and should achieve a short runtime, it needs to add sets simultaneously. Hence, it has to be ensured that at least two sets that are added concurrently do not cover the same elements. Thus, the main goals are in this BEEPING model:

1. To estimate the number of uncovered neighbors correctly.

2. To avoid too many sets containing the same elements being added to the solution concurrently.

The authors us randomization and present a random mechanism to achieve the desired properties. For simplicity, they present the following definition. For any $i = k, \ldots, 0$ we define

$$\Delta_i := \frac{\Delta}{\Delta^{\frac{i}{k}}} := \left(\Delta^{\frac{1}{k}}\right)^{k-i}. \tag{4.2}$$

There are two assumptions about the degree. First, assume that Δ_i is an integer for any choice of i. This means that Δ should be a multiple of $\Delta^{\frac{1}{k}}$. With this help,

there is no need to carry rounding artifacts through our equations. Second, assume w.l.o.g. that $\Delta > 4k$. The largest meaningful value for k is $O(\log \Delta)$. For bigger values, the term $O(\log^2 \Delta)$ dominates the approximation ratio given in Theorem 4.16. As $\Delta \in \omega(\log \Delta)$, the assumption holds for instances of non-constant degree. Instances of degree $\Delta \in O(1)$ admit a trivial constant-factor approximation by adding all sets. Like this the result remains general.

The algorithm runs in k phases. In phase i, all sets that cover *approximately* Δ_i elements try to add themselves to the solution. At the beginning of each *phase*, the sets and elements do the following:

- Each *set* s draws a geometric random variable X_s with parameter $1 - \frac{1}{\Delta^{\frac{1}{k}}}$. Values bigger than $4k$ are rounded down to $4k$, so $X_s \in [0, 4k]$ always.
- Each uncovered *element* u draws $4k$ variables $Y_u^1, \ldots, Y_u^{4k}$ with $Y_u^\ell \in \{0, 1\}$ and $Pr[Y_u^\ell = 1] = \left(\frac{1}{\Delta_i}\right)$ independently and uniformly at random. If at least one of these variables is positive, we say that u is *active*. As it is shown later, this ensures that each set with Δ_i uncovered elements has roughly $4k$ active elements in expectation.

The variables are fixed for the duration of the phase.

Not all of the sets with active elements in a given phase may be simultaneously added to the solution, because this would not lead to a good approximation. A lot of these sets could cover the same active elements. This is the reason why the algorithm must add them carefully and update the number of active elements after each addition. The phases are divided into $4k + 1$ rounds. Here only some sets try to add themselves. So the geometric distribution comes into play. Each set has precisely one round $\Phi_s \in [0, 4k]$ where it wants to enter the solution. The round is determined by $\Phi_s := 4k - X_s$. In the other rounds, the set will be idle and it will not try to add itself. This means that in the first couple of rounds, only a few sets try to add themselves and only cover elements contained in many sets. In the later rounds, almost all sets try to add themselves. The particular choice of parameters ensures that in each round, only $\tilde{O}(\Delta^{\frac{1}{k}})$ sets try to cover the same element simultaneously in expectation.

The authors focus on the order of events of a single round, where a single round consists of $4k + 1$ slots. In the first $4k$ slots, the sets try to estimate the number of their active elements. In this time, the sets only listen and count the number of beeps that they hear. Each active *element* u that has not been covered beeps in slot $\ell \in [1, 4k]$ if it has drawn $Y_u^\ell = 1$ in the beginning. It remains idle otherwise. The authors consider a set that has at least Δ_i uncovered elements. The choice of parameters implies that in each slot, there is at least one active element that sends a Beep *in expectation*. After these $4k$ steps, the sets decide to enter the solution. Each *set* s with $\Phi_s = j$ that received a Beep in at least $3k$ slots adds itself to the solution and beeps. In the final slot of the round, the elements listen for beeps. If they hear a beep, they set themselves as covered. In the last phase, it holds $\frac{1}{\Delta_k} = 1$.

Thus, all uncovered nodes become active and beep in all $4k$ slots. Hence, even if there is only one uncovered element, the neighboring sets hear a BEEP in $4k$ slots. As each set eventually tries to add itself, all elements will be covered.

Each element counts how many sets want to cover this element. The element sends this value to all of these sets. Hence, each set receives a value from all its elements that are not covered. The sets locally compute the median of these values. They pick its inverse as their probability. This calculation needs each element and set to send and receive *distinct* messages of size $O(\log \Delta)$ to and from all of their neighbors, because all values are smaller than Δ and require $O(\log \Delta)$ bits to be encoded. This is trivial to implement in the CONGEST model. But in the BEEPING model, this calculation cannot be done so easily because the communication is restricted. First, it would need at least $O(\log \Delta)$ slots to submit even a single value, because only one bit per round can be transmitted. Hence, a simple simulation cannot achieve a runtime polynomial in k. Second, it cannot receive distinct values from different neighbors in the BEEPING model.

Figure 4.4 presents the pseudocode for the algorithm. The code obtains the following notation. If an element wants to listen to BEEPS, it calls the method LISTEN(). This method either returns BEEP if at least one neighboring node sent out a BEEP or $\perp$ otherwise. $B(p)$ and $Geo(p)$ are used to refer to the Bernoulli and geometric distribution, respectively. The variables store the aforementioned random variables. The nodes have little additional state information. Each set s has a variable $S_s \in \{0, 1\}$ that is 1 if s is part of the solution set and 0 otherwise. This variable is initially set to 0. In addition, each set has an array $A \in \{\perp, \text{BEEP}\}^{4k}$ of size $4k$. This array is used the count the beeps of neighboring elements received in one round. After each round, this array is reset. Thus, the total memory of a set is only $O(k)$ bits. The elements have the variable $C_e \in \{0, 1\}$. This variable takes value 1 if the element is covered and 0 otherwise.

The authors present in [18] the following results that will be relevant for our SetCover algorithm in chapter 10. The runtime of $O(k^3)$ comes trivially. The algorithm is structured in k phases. Each phase is structured in $4k + 1$ *rounds*. In addition, each round again consists of $4k + 1$ slots. This brings the total runtime to $O(k^3)$. Since the runtime is deterministic, it is only important to prove the expected approximation ratio.

Lemma 4.17. The algorithm outputs a $O\left(\Delta^{\frac{3}{k}} \log^2 \Delta\right)$-approximate solution in expectation.

An element can be covered in the last slot of a round. There are $k + 1$ phases and $4k + 1$ rounds in a phase. Thus, there are $\tau := 4(k + 1)^2 - 3$ rounds in total. For this analysis, the authors write each round as a tuple (i, j) where $i \in [0, k]$ is the phase and $j \in [0, 4k]$ is the round of that phase. Thus, when they write *round* (i, j), they mean the j^{th} round of the i^{th} phase. i and j correspond to the loop counters in the pseudocode. The authors choose this notation because the phase i is very important for calculating the probabilities. In the following part, consider

the round $(0,0) \leq (i,j) \leq (k,4k)$ in which element $u \in V_{\mathscr{U}}$ is covered. Because all elements are eventually covered (in the last phase, all uncovered elements beep until one of their neighboring sets adds itself). Hence, this phase is well-defined. For each element covered by the algorithm, define the random variables $\eta_{(i,j)}(u) \in [1,\Delta]$ and $\mu_{(i,j)}(u) \in [1,\Delta]$. Define $\eta_{(i,j)}(u)$ to be the ratio between the *best* set in u's neighbor and the worst set picked by the algorithm, i.e., the ratio by which the choice of the algorithm differs from the greedy solution. Further, let $\mu_{(i,j)}(u)$ be the random variable that denotes the number of candidates covering $u \in V_{\mathscr{U}}$ in the round where it is first covered.

Moreover, define the functions $c_{min}^{(i,j)}(u) \in [\frac{1}{\Delta},1]$ and $c_{max}^{(i,j)}(u) \in [\frac{1}{\Delta},1]$ for each covered element $u \in V_{\mathscr{U}}$. Let $N_u \subset V_S$ denote the sets that can cover u. Let N_s denote the elements contained in some set $s \in V_{\mathscr{S}}$. Assume u is covered by sets $C_{(i,j)}(u) \subseteq N_u$. These sets add themselves simultaneously. Further, let $d_{(i,j)}(s)$ for $s \in N_u$ be the *span* of set s, i.e., the number of uncovered elements neighboring s. Based on this, one can define $c_{max}^{(i,j)}(u)$ based on the set with *fewest* uncovered elements. The set that deviates the most from whatever set the greedy solution would have picked. We have:

$$c_{max}^{(i,j)}(u) = \max_{s \in C_{(i,j)}(u)} \frac{1}{d_{(i,j)}(s)} \tag{4.3}$$

The value $c_{min}^{(i,j)}(u)$ is determined by the set in u's neighborhood with the biggest possible span. The set that the greedy solution would have picked. This set may not be part of $C_{(i,j)}(u)$. Formally, the authors obtain the following:

$$c_{min}^{(i,j)}(u) = \min_{s \in N_u} \frac{1}{d_{(i,j)}(s)} \tag{4.4}$$

On the other hand, by the definition of $\eta_{(i,j)}(u)$, it holds:

$$c_{max}^{(i,j)}(u) = \eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u) \tag{4.5}$$

Finally, they define the following values:

1. Let $S_{(i,j)}$ be set of candidates that add themselves to S in round i.

2. For each $s \in S_{(i,j)}$ let $U_{(i,j)}(s) \subset N_s$ denote the uncovered elements in s. Note that $d_{(i,j)}(s) := |U_{(i,j)}(s)|$, i.e., the cardinality of $U_{(i,j)}(s)$ is the set's span.

3. Let $U_{(i,j)} := \bigcup_{s \in S_{(i,j)}} U_{(i,j)}(s)$ denote the set of elements that are uncovered at the start of round (i,j) and are covered in this round.

Given these definitions, the number of sets that are added to the solution in round

(i, j) can be bounded as follows:

$$|S_{(i,j)}| \leq \sum_{s \in S_{(i,j)}} \frac{d_{(i,j)}(s)}{d_{(i,j)}(s)} \leq \sum_{s \in S_{(i,j)}} |U_{(i,j)}(s)| \frac{1}{d_{(i,j)}(s)} \tag{4.6}$$

$$\triangleright \textit{ As } s \textit{ covers exactly } d_{(i,j)}(s) \textit{ elements.} \tag{4.7}$$

$$\leq \sum_{s \in S_{(i,j)}} \sum_{u \in U_{(i,j)}(s)} c_{max}^{(i,j)}(u) \tag{4.8}$$

$$\triangleright \textit{ As } 1/d_{(i,j)}(s) \leq c_{max}^{(i,j)}(u) \textit{ per definition.} \tag{4.9}$$

$$= \sum_{u \in U_{(i,j)}} c_{max}^{(i,j)}(u) \cdot \mu_{(i,j)}(u) \tag{4.10}$$

$$\triangleright \textit{ As each } u \in U_{(i,j)} \textit{ is covered by } \mu_{(i,j)}(u) \textit{ sets.} \tag{4.11}$$

$$\leq \sum_{u \in U_{(i,j)}} c_{min}^{(i,j)}(u) \cdot \eta_{(i,j)}(u) \cdot \mu_{(i,j)}(u) \tag{4.12}$$

$$\triangleright \textit{ As } c_{max}^{(i,j)}(u) := \eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u). \tag{4.13}$$

Consider the expected value of $|S_{(i,j)}|$. With the linearity of expectation and the inequality above, we have:

$$E\left[|S_{(i,j)}|\right] \leq \sum_{u \in V_{\mathscr{U}}} u \in U_{(i,j)} \cdot E[\mu_{(i,j)}(u) \cdot \eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u)] \tag{4.14}$$

$$\leq \sum_{u \in V_{\mathscr{U}}} u \in U_{(i,j)} \cdot E[\mu_{(i,j)}(u)] \cdot E[\eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u)] \tag{4.15}$$

The second line comes from the fact that the random events that determine $\eta_{(i,j)}(u)$ and $\mu_{(i,j)}(u)$ are independent. The authors bound $E[\eta_{(i,j)}(u) c_{min}^{(i,j)}(u)]$ and $E[\mu_{(i,j)}(u)]$ separately. In particular, for $\mu_{(i,j)}(u)$, we show the following:

Lemma 4.18. For all elements $u \in V_{\mathscr{U}}$ and all round (i, j), it holds:

$$E[\mu_{(i,j)}(u)] \leq 3\Delta^{\frac{1}{k}} \tag{4.16}$$

The proof is based on the specific properties of the geometric distribution and can be found in Section 4.5. It holds for the difference between the best and the worst set:

Lemma 4.19. Let $k \geq 3$. Then, for every element u and round (i, j) in which u is covered, it holds

$$E[\eta_{(i,j)}(u) c_{min}^{(i,j)}(u)] \leq 25 \cdot \log \Delta \cdot \left(\Delta^{\frac{1}{k}}\right)^2 E[c_{min}^{(i,j)}(u)] \tag{4.17}$$

The proof can be found in Section 4.5. Putting these two insights back in the formula, we have:

$$E\left[|S_{(i,j)}|\right] \leq \sum_{u \in V} u \in U_{(i,j)} \cdot E[\mu_{(i,j)}(u)]E[\eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u)] \qquad \triangleright \textit{By Equation (4.15)}$$

(4.18)

$$\leq \left(3\Delta^{\frac{1}{k}}\right) \sum_{u \in V} u \in U_{(i,j)} \cdot E[\eta_{(i,j)}(u) \cdot c_{min}^{(i,j)}(u)] \qquad \triangleright \textit{By Lemma (4.18)}$$

(4.19)

$$\leq \left(3\Delta^{\frac{1}{k}}\right) \left(25\left(\sqrt[k]{\Delta}\right)^2 \log \Delta\right) \sum_{u \in V} u \in U_{(i,j)} E[c_{min}^{(i,j)}(u)] \qquad \triangleright \textit{By Lemma (4.19)}$$

(4.20)

$$\leq \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \sum_{u \in V} u \in U_{(i,j)} E[c_{min}^{(i,j)}(u)] \tag{4.21}$$

Recall that each element gets covered eventually. Hence, eventually, each u gets assigned as cost $c_{min}^{(i,j)}(u)$. For each $u \in V_{\mathscr{U}}$ the expected cost are defined as:

$$c_{min}(u) := \sum_{i=0}^{k} \sum_{j=0}^{4k} u \in U_{(i,j)} c_{min}^{(i,j)}(u) \tag{4.22}$$

Next, we can use the linearity of expectation to sum over all k^2 rounds and get:

$$E\left[|S|\right] = \sum_{i=0}^{k} \sum_{j=0}^{4k} E[|S_{(i,j)}|] \qquad \triangleright \textit{Linearity of Exp.}$$

(4.23)

$$\leq \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \sum_{i=0}^{k} \sum_{j=0}^{4k} \sum_{u \in V_{\mathscr{U}}} u \in U_{(i,j)} E[c_{min}^{(i,j)}(u)] \qquad \triangleright \textit{Inequality (4.21)}$$

(4.24)

$$= \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \sum_{u \in V_{\mathscr{U}}} E[c_{min}(u)] \qquad \triangleright \textit{Law of Total Exp.}$$

(4.25)

It remains to bound the *expected* cost assigned to an element when it is covered. By the analysis of the greedy algorithm by Jia et al., it holds that:

Lemma 4.20 (Lemma 3.5 in [22])**.** Let $|S_{OPT}|$ be the size of the optimal solution, then it holds:

$$\sum_{u \in V_{\mathscr{U}}} c_{min}(u) \leq H_{\Delta}|S_{OPT}| \leq \log \Delta |S_{OPT}| \tag{4.26}$$

Here, the term H_{Δ} denotes the Δ^{th} harmonic number.

Fix any execution $\mathfrak{A}$ of the algorithm, i.e., assign each element a round in which it is covered. For all possible executions, the maximal cost of $H_\Delta|S_{OPT}|$ is assigned. The proof exploits that c_{min} can always be bounded by the span of a set from the optimal solution.

Consider a set $s \in S_{OPT}$ of degree Δ_s. This set may not be a part of the solution computed by the algorithm. The authors use it to bound the assigned cost. Consider only the rounds in which some neighbor of s is covered in the following and ignore all the rounds in between. Assume that there are r distinct rounds $(i_1, j_1), \ldots, (i_r, j_r)$ in which neighbors are covered. For simplicity enumerate these rounds as $1, \ldots, r$. The concrete indices of the phase and round in which an element is covered are *not* important for this proof. Define the values $\delta_1, \ldots, \delta_r$ which denote the number of neighbors covered in the corresponding round. More concrete, in the l^{th} round where some elements of s are covered, the algorithm covers exactly $\delta_l \geq 1$ elements. All these values are only determined by $\mathfrak{A}$.

In the first round, where any element of s is covered, it is assigned a cost of at most $\frac{1}{\Delta_s}$. If there is a set with a bigger span, the cost can only get smaller. In the second round, where any neighbor of s is covered, it is assigned a cost of at most $\frac{1}{\Delta_s - \delta_1}$. δ_1 counts how many elements were covered in the first round. Thus, s must have a span of $\Delta_s - \delta_1$ because, per definition. None of the other neighbors of s was covered in between. This argument can be repeated inductively, such that the total cost after the r^{th} round is bounded by

$$\sum_{u \in N_s} E[c_{min}(u) \mid \mathfrak{A}] \leq \sum_{l=1}^{r} \frac{1}{\Delta_s - \delta_l} \tag{4.27}$$

Seeking an upper bound, let $u_1, \ldots, u_{\Delta_s}$ be elements such that u_ℓ is covered before or in the same round as $u_{\ell+1}$. The cost assigned to u_ℓ is bounded by $\frac{1}{\Delta_s - (\ell-1)}$. Hence, the maximal cost that a node of the optimal solution can assign is bounded by

$$\begin{aligned}
\sum_{u \in N_s} E[c_{min}(u) \mid \mathfrak{A}] &\leq \sum_{l=1}^{r} \frac{1}{\Delta_s - \delta_l} && (4.28)\\
&\leq \sum_{\ell=1}^{\Delta_s} \frac{1}{\Delta_s - (\ell - 1)} && (4.29)\\
&\leq H_{\Delta_s} \leq H_\Delta && (4.30)
\end{aligned}$$

Because there are $|S_{OPT}|$ sets in the optimal solution, which all assign cost H_Δ, the lemma follows for assignment $\mathfrak{A}$. Because this bound holds for *all* possible executions of the algorithm, the law of total expectation yields to the lemma. Putting

everything together, we get:

$$E[|S|] \leq \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \sum_{u \in V_{\mathscr{U}}} E[c_{min}(u)] \tag{4.31}$$

$$\leq \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \cdot H_\Delta \cdot |S_{OPT}| \qquad \triangleright \textit{Lemma 4.20} \tag{4.32}$$

$$\leq \left(75\Delta^{\frac{3}{k}} \log \Delta\right) \cdot \log(\Delta) \cdot |S_{OPT}| \tag{4.33}$$

$$\leq \left(75\Delta^{\frac{3}{k}} \log^2 \Delta\right) \cdot |S_{OPT}| \tag{4.34}$$

$$\in O\left(\Delta^{\frac{3}{k}} \log^2(\Delta) |S_{OPT}|\right) \tag{4.35}$$

This proves the main theorem.

Proof of Lemma 4.18

Further, the authors show that for any node $u \in V_{\mathscr{U}}$, it holds that the probability that u is covered by more than t sets simultaneously is exponentially declining in t. This is independent of the phase and round in which u is covered. One main result is:

Lemma 4.21. For any value $t > 1$, any element $u \in V$ and any round (i, j) in which u is covered, it holds:

$$\mathbf{Pr}\left[\mu_{(i,j)}(u) = t\right] \leq \max\{e^{-t/4}, 2\left(1 - 1\Delta^{\frac{1}{k}}\right)^{t-1}\} \tag{4.36}$$

The authors use a result by Miller et al. [33]. This result needs to be adapted for geometric values. In [33], they proved that the number of candidates that pick the earliest possible wake-up time is $\Delta^{\frac{1}{k}}$ in expectation. For each node $u \in V_{\mathscr{U}}$, the authors of [18] consider the *phase* in which u is covered. Every element is *always* covered as in the last round of the last phase, all remaining sets that cover at least one element simply add themselves to the solution.

The authors focus only on a single phase i. For simplicity, refer to the number of sets that cover u simply as $\mu(u)$ instead of $\mu_{(i,j)}(u)$) during this proof. Since one focuses on a single fixed phase, one can just write *round* j instead of *round* (i, j). Whenever it is written a round $j \in [0, 4k]$, it means *round* j *of phase* i.

Throughout the proof, divide the sets into two subsets.

1. First, let N_u be the set of sets in u's neighborhood, i.e., the sets that can cover u. Call these the *neighboring candidates*.

2. Second, let $NN := V_S \setminus N_u$ be all other sets. Call these the *non-neighboring candidates*.

Define the wakeup time Φ_s of each $s \in V_S$ as $\Phi_s := 4k - X_s$. Next, consider the concrete round j in which u is covered. It is important to distinguish between round

$j = 0$ and all other rounds. First, consider the elements covered in round $j = 0$. In this case, the values X_s of *all* candidates that cover u must be bigger than or equal to $4k$ and before being rounded down. Otherwise, if it held $X_s < 4k$, the value $\Phi_s := 4k - X_s$ would be strictly positive, and the corresponding set would wake up in a later round. In the following, call a set $s \in V_{\mathscr{S}}$ *early* if it holds $X_s \geq 4k$. Given this definition, it holds:

Lemma 4.22. Denote $\mathscr{E}_u$ as the number of early sets covering u. Then it holds:

$$E[\mathscr{E}_u] \leq \frac{1}{\Delta^2} \tag{4.37}$$

For a single candidate $s \in N_u$, the probability to be *early* is at most

$$\begin{aligned} Pr[s \text{ is early}] &= \sum_{i=0}^{\infty} Pr[X_s = 4k + i] && (4.38)\\ &= \sum_{i=0}^{\infty} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{4k+i} \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right) && (4.39)\\ &\leq \sum_{i=0}^{\infty} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{4k} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{i} && (4.40)\\ &= \frac{1}{\Delta^4} \sum_{i=0}^{\infty} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{i} \leq \frac{2}{\Delta^4} && (4.41) \end{aligned}$$

This follows directly from the definition of the geometric distribution. As there are *at most* Δ neighboring candidates that cover u, the expected number of early candidates is at most $\frac{2}{\Delta^3}$ as it holds that:

$$E[\mathscr{E}_u] = \sum_{s \in N_u} Pr[s \text{ is early}] = \sum_{s \in N_u} \frac{2}{\Delta^4} \leq \frac{2\Delta}{\Delta^4} = \frac{2}{\Delta^3} \tag{4.42}$$

As $\Delta \geq 2$ this further simplifies to $\frac{1}{\Delta^2}$. Given the fact that all set pick their wake-up time independently, the Chernoff bound implies the following:

Lemma 4.23. Suppose u gets covered in the first round of the phase, then

$$Pr[\mu(u) \geq t \mid u \text{ is covered in round } 0] \leq e^{-\frac{t}{4}} \tag{4.43}$$

Since all candidates pick their wake-up time independently and $\mu(u)$ is exactly the number of early candidates, i.e., it holds $\mu(u) = \mathscr{E}_u$, on can make use of the

Chernoff bound to show that:

$$
\begin{aligned}
Pr[\mu(u) \geq 4t' \mid u \textit{ is covered in round } 0] &= Pr[\mathscr{E}_u \geq 4t'] && (4.44)\\
&= Pr[\mathscr{E}_u \geq 4t'\Delta^2 \frac{1}{\Delta^2}] && (4.45)\\
&\leq Pr[\mathscr{E}_u \geq 4 \cdot t' \cdot \Delta^2 E[\mathscr{E}_u]] && \triangleright \textit{By Lemma 4.22} \quad (4.46)\\
&\leq Pr[\mathscr{E}_u \geq \left(1 + 3 \cdot t' \cdot \Delta^2\right) E[\mathscr{E}_u]] && (4.47)\\
&\leq e^{\frac{-3t' \cdot \Delta^2}{\Delta^2 \cdot 3}} \leq e^{-t'} && \triangleright \textit{Chernoff bound} \quad (4.48)
\end{aligned}
$$

The lemma follows by substituting $t = t'/4$. Having dealt with round 0, assume that u gets covered in any other round $j > 0$. First, fix all of the random decisions made by the algorithm **except** the decision of the neighboring candidates. More concrete, condition on the following three variables:

1. **The wake-up times of the non-neighboring sets Z_{NN}.**

   It holds $Z_{NN} := (\phi_{s_1}, \ldots, \phi_{s_{|NN|}})$, such that the non-neighboring set $s_l \in NN$ wakes up in round ϕ_{s_l}.

2. **The slots Z_U in which the uncovered elements send a Beep.**

   This random variable contains all random choices made by the uncovered elements, i.e., $Z_U := \{(Y^1_{u'}, \ldots, Y^{4k}_{u'}) \mid u' \in V_{\mathscr{U}}\}$. Here, the variable $Y^\ell_{u'}$ denotes whether u' is active and beeps in slot ℓ. Based on this, we can define the *number* of slots in which a set $s \in NN$ receives a Beep from an element in subset $U \subseteq V_{\mathscr{U}}$ as follows:

   $$\textsc{Slots}(s, U) := \left|\left\{\ell \in [1, 4k] | \exists u' \in U \cap N_s : Y^\ell_{u'} = 1\right\}\right| \qquad (4.49)$$

3. **The initial set of uncovered elements $\mathscr{U}_0$.**

   This is the set of uncovered elements in round 0 of phase i. Note that this set is determined in the previous phases. We do not make any statements about its size or the nodes it contains.

One will see that the decision, if a neighboring candidate of u adds itself to the solution, depends on *solely* that candidate's wake-up time if we fix all these choices. Note that a set adds itself in round j if and only if a) it wakes up in round j **and** b) hears a Beep in at least $3k$ slots of round j. The authors show that the random choices of the other nodes define a round $\rho \in [-1, 4k]$ such that in all rounds prior to ρ, s will hear at least $3k$ Beeps and in all later round it will not (and therefore will not add itself). If there is no round where s hears more than $3k$ Beeps, define $\rho := -1$. Hence, ρ is the last possible round where a candidate may add itself to the solution. If it wakes up after this round, i.e., if it holds $4k - X_s > \rho$, it will not

add itself because it will not hear enough Beeps in that round. If it wakes up in or before this round, i.e., if it holds $4k - X_s \leq \rho$, then it wakes up in a round where it receives sufficiently many Beeps. Hence, in the latter case, the concrete round in which it adds itself is *only* determined by the geometric distribution. This intuitive description of ρ is now captured in the following claim: Given $(Z_{NN}, Z_U, \mathscr{U}_0)$ for each neighboring set $s \in N_u$ there is a value $\rho_s \in [-1, 4k]$ that denotes the last round in which s can cover u (or takes value $\rho_s = -1$ if there is no such round). We have

$$Pr[s \text{ covers } u \text{ first } \mid \Phi_s > \rho_s] = 0. \tag{4.50}$$

The main idea behind the construction is that the *fixed* values $\mathscr{U}_0$, Z_{NCC}, and Z_U can define the sets $\mathscr{A}_0 \subset V_{\mathscr{S}}$ that add themselves to the solution in round 0. Given Z_{NN}, one can see which sets wake up and count the Beeps of their uncovered elements. The Beeps are based on $\mathscr{U}_0$ and Z_U, so one can see if enough elements beep in distinct slots. More concretely, the set must receive a Beep in at least $3k$ slots, so it holds

$$\mathscr{A}_0 := \{s \in NN_u \mid \phi_s = 0 \wedge (|\textsc{Slots}(s, \mathscr{U}_0)| \geq 3k)\} \tag{4.51}$$

Given $\mathscr{A}_0$, i.e., the sets that add themselves in round 0, one can compute $\mathscr{U}_1 \subseteq \mathscr{U}_0$, i.e., all uncovered elements in round 1. Then, by repeating the construction, we can use $\mathscr{U}_1$, Z_{NCC}, and Z_U to determine $\mathscr{A}_1$ and therefore $\mathscr{U}_2$ and so on. This can be continued until round τ by the following recursive formulas:

$$\mathscr{A}_t = \{s \in NN_u \mid \phi_s = 0 \wedge (|\textsc{Slots}(s, \mathscr{U}_t)| \geq 3k)\} \tag{4.52}$$

and

$$\mathscr{U}_t = \left\{ u' \in \mathscr{U} \setminus \{u\} \mid \exists s \in \bigcup_{i=0}^{t-1} \mathscr{A}_i \right\} \tag{4.53}$$

This yields the uncovered elements $\mathscr{U}_0, \ldots, \mathscr{U}_\tau$. For each candidate $s \in N_u$, one can identify the last round ρ_s where s receives enough Beeps to add itself or whether there is no such round. Define:

$$\rho_s := \begin{cases} _{0 \leq \tau \leq 4k}\{\textsc{Slots}(s, \mathscr{U}_\tau) \geq 3k\} & \text{if } \exists \tau \in [0, 4k] : \{\textsc{Slots}(s, \mathscr{U}_\tau) \geq 3k\} \\ -1 & \text{else} \end{cases} \tag{4.54}$$

Hence, if the wake-up time is later than ρ_s, s will not hear sufficiently Beeps. This holds for $\rho_s = -1$ since $\Phi - X_s \geq 0$.

Given this observation, one can map each outcome of Z_{NCC}, Z_U, and $\mathscr{U}_0$ to a collection of thresholds $\overline{\rho} := (\rho_1, \ldots, \rho_{||N_u})$, one for each neighboring set, with the properties mentioned above. Now condition on $\overline{\rho}$. The result is then similar to Miller et al. in [33] and looks as follows:

Lemma 4.24. For any possible realization of thresholds $\overline{\rho}$ one obtains:

$$Pr[\mu(u) = t \mid \overline{\rho}] \leq 2\left(1 - 1/\Delta^{\frac{1}{k}}\right)^{t-1} \tag{4.55}$$

For each neighboring set $s \in N_u$, define the adapted wake-up time as:

$$\Phi'_s := \begin{cases} (\Phi - X_s) & \textit{if } (\Phi - X_s) \leq \rho_s \\ \infty & \textit{else} \end{cases} \tag{4.56}$$

Here X_s is the geometric random variable drawn to determine the wake-up time. Assume that u has exactly Δ_u neighboring sets, all of which have a well-defined wake-up time (which may be infinite). The authors order the these adapted wakeup times $\Phi'_{(1)}, \ldots, \Phi'_{(\Delta_u)}$ such that $\Phi'_{(1)}$ is the earliest wakeup and $\Phi'_{(\Delta_u)}$ is the last. For each of these ordered wake-up times $\Phi_{(\ell)}$ we define

- $s_{(\ell)}$ to be the candidate that achieves this time
- $X_{(\ell)}$ is the variable drawn by this set, and
- $\rho_{(\ell)}$ is the threshold.

Given that $\Phi_{(\ell)} \leq \infty$, it holds:

$$\Phi'_{(\ell)} := \Phi - X_{(\ell)} \tag{4.57}$$

So the variable $X_{(1)}$ is *not* the smallest variable that any neighboring candidate drew. Instead, it is the smallest variable by any neighboring candidate *with finite wake-up time*. Given this definition, the authors claim the following: The candidate $s_{(1)}$ that achieves $\Phi'_{(1)}$ covers u if and only if $\Phi'_{(1)} \leq \infty$. This claim can be verified with the two possibilities. If $\Phi'_{(1)} \leq \infty$, then $\Phi'_{(1)}$ is still the earliest round where some neighborhood candidate tries to add itself and is still a candidate. This follows from the definition of $\Phi'_{(1)}$. More concrete, the corresponding set $s_{(1)}$ wakes up in round $\Phi'_{(1)}$ and — since $\Phi'_{(1)} \leq \rho_{(1)}$ — counts more than $3k$ Beeps in this round. Hence, $s_{(1)}$ must cover u because no candidate could have done it before. Otherwise, if $\Phi'_{(1)} = \infty$, then it holds that $\Phi - X_{(1)} \geq \rho_{(1)}$. This implies that $\Phi - X_{(\ell)} \geq \rho_{(\ell)}$ for all other candidates as well. This follows from two facts. First, $\Phi'_{(1)}$ is the smallest wake-up time by definition. So all others must be infinity, too. Second, $\rho_{(\ell)}$ is at most $4k$ and finite by definition. In this case, *all* potential candidates do not cover enough elements when they wake up. This follows from the definition of each $\rho_{(\ell)}$. Hence, in this case, u is not covered by any candidate in this phase.

Hence, for $\mu(u) = t$, the t smallest values must all be equal and non-infinity. More formally:

$$Pr[\mu(u) = t] = Pr[\Phi'_{(1)} = \cdots = \Phi'_{(t)} \mid \Phi_{(1)} \leq \infty] \tag{4.58}$$

Next, the authors make use of the fundamental calculations based on Miller et al.'s proof for exponential random variables. One must adapt them to geometric

variables bounded by some maximal value. First, we deal with the bounding. Recall that all of our variables must be smaller than $4k$. Otherwise, u would have been covered in round 0. The Law of Total Probability implies:

$$Pr[\mu(u) = t \mid X_{(1)}, \ldots, X_{(\Delta_u)} \leq 4k] \leq \frac{Pr[\mu(u) = t]}{Pr[X_{(1)}, \ldots, X_{(\Delta_u)} \leq 4k]} \quad \triangleright \textit{Law of Total Prob.} \tag{4.59}$$

$$\leq \frac{Pr[\mu(u) = t]}{1 - Pr[\mathscr{E}_u \geq 1]} \quad \triangleright \textit{As } i \textit{ is early iff } X_i \geq 4k \tag{4.60}$$

$$\leq \frac{Pr[\mu(u) = t]}{1 - Pr[\mathscr{E}_u \geq \Delta^2 E[\mathscr{E}_u]} \quad \triangleright \textit{By Lemma 4.22} \tag{4.61}$$

$$\leq \frac{Pr[\mu(u) = t]}{\left(1 - \frac{1}{\Delta^2}\right)} \quad \triangleright \textit{Markov ineq.} \tag{4.62}$$

$$\leq 2Pr[\mu(u) = t] \quad \triangleright \textit{Using } \Delta > 2 \tag{4.63}$$

Thus, in the following, the authors analyze the variables as if they were not bounded. In the end, they multiply the final result by 2 to get the desired bound.

Next, the authors show the following claim: For any $0 \leq x \leq \rho_{(\ell)}$, it holds:

$$Pr[\Phi'_{(\ell)} < x \mid \Phi'_{(\ell)} \leq x] = \left(\frac{1}{\Delta^{\frac{1}{k}}}\right) \tag{4.64}$$

Let $x := \rho_{(\ell)} - x'$, then it holds by the definition of $\Phi'_{(\ell)}$:

$$Pr[(\Phi - X_{(\ell)}) \leq \rho_{(\ell)} - x'] = Pr[-X_{(\ell)} \leq \rho_{(\ell)} - \Phi - x'] \tag{4.65}$$

$$= Pr[X_{(\ell)} \geq (\Phi - \rho_{(\ell)}) + x'] \tag{4.66}$$

For any $x \leq \rho_{(\ell)}$ we have that the wake-up time is only determined by the parameter of the geometric distribution. This is:

$$Pr[\Phi'_{(\ell)} < \rho_{(\ell)} - x' \mid \Phi'_{(\ell)} \leq \rho_{(\ell)} - x'] = Pr[(\Phi - X_{(\ell)}) < \rho_{(\ell)} - x' \mid (\Phi - X_{(\ell)}) \leq \rho_{(\ell)} - x'] \tag{4.67}$$

$$= Pr[X_{(\ell)} > (\Phi - \rho_{(\ell)}) + x' \mid X_{(\ell)} \geq (\Phi - \rho_{(\ell)}) + x'] \tag{4.68}$$

One can use the fact that the geometric distribution is memoryless and obtain:

$$Pr[\Phi'_{(\ell)} < x \mid \Phi'_{(\ell)} \leq x] = Pr[X_{(\ell)} > (\Phi - \rho_{(\ell)}) + x' \mid X_{(\ell)} \geq (\Phi - \rho_{(\ell)}) + x'] \tag{4.69}$$

$$= Pr[X_{(\ell)} > y \mid X_{(\ell)} \geq y] \tag{4.70}$$

$$\triangleright \textit{ Substituting } y := (\Phi - \rho_{(\ell)}) + x' \tag{4.71}$$

$$= Pr[X_{(\ell)} > 0] = \left(\frac{1}{\Delta^{\frac{1}{k}}}\right) \tag{4.72}$$

Therefore, one can conclude that for two consecutive $\Phi_{(i)}$ and $\Phi_{(i+1)}$, we have:

$$Pr[\Phi'_{(\ell)} < x \mid \Phi'_{(\ell+1)} = x] = Pr[\Phi'_{(\ell)} < x \mid \Phi'_{(\ell)} \leq x] \quad \triangleright As\ \Phi'_{(\ell)} \leq \Phi'_{(\ell+1)} \tag{4.73}$$

$$= \left(\frac{1}{\Delta^{\frac{1}{k}}}\right) \tag{4.74}$$

Hence, for the opposite event, we have:

$$Pr[\Phi'_{(\ell)} = x \mid \Phi'_{(\ell+1)} = x] = \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right) \tag{4.75}$$

Finally, condition on $\Phi'_{(t)} = \tau$ for a round $0 \leq \tau \leq 4k$. Using the chain rule of conditional probability, we obtain that:

$$Pr[\Phi'_{(1)}, \ldots, \Phi'_{(t)} = \tau \mid \Phi'_{(t)} = \tau] = \prod_{i=1}^{t-1} Pr[\Phi'_{(\ell)} = \tau \mid \Phi'_{(\ell+1)} = \tau, \ldots, \Phi'_{(t)} = \tau] \tag{4.76}$$

$$= \prod_{i=1}^{t-1} Pr[\Phi'_{(\ell)} = \tau \mid \Phi'_{(\ell+1)} = \tau] \tag{4.77}$$

$$\leq \prod_{i=1}^{t-1} \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right) = \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right)^{t-1} \tag{4.78}$$

This is independent of the actual round. Thus, the law of total probability yields the result. We have:

$$Pr[\mu(u) = t] := \sum_{\tau=1}^{4k} Pr[\Phi'_{(t)} = \tau] Pr[\Phi'_{(1)}, \ldots, \Phi'_{(t)} = \tau \mid \Phi'_{(t)} = \tau] \tag{4.79}$$

$$= \sum_{\tau=1}^{4k} Pr[\Phi'_{(t)} = \tau] \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right)^{t-1} \tag{4.80}$$

$$= \left(1 - \frac{1}{\Delta^{\frac{1}{k}}}\right)^{t-1} \tag{4.81}$$

As the concrete values of the ρ's are immaterial, the lemma follows by the law of total probability.

Now conclude the proof and calculate the expected value of μ. Given the previous work, the calculation of the expected value is straightforward as it is shown below). [Proof of Lemma 4.18] Implied by the previous lemmas, the authors split the proof into two parts. First, we consider the *easy case*, when u is covered in the first round of a phase. Lemma 4.22 provides the upper bound for the number of additional nodes that also wake up in round 0 as follows:

$$E\left[\mu(u) \mid u \textit{ covered in round } 0\right] \leq \frac{1}{\Delta^2} \leq 1. \tag{4.82}$$

Second, if u is covered in any later round of a phase, one needs to make use of Lemma 4.21. The lemma shows that the number of sets is geometrically distributed with additional factors. Using the limit of the geometric series, one can show that:

$$E\left[\mu(u) \mid u \textit{ not covered in round } 0\right] \leq \sum_{t=2}^{\infty} t \cdot Pr[\mu = t \mid u \textit{ not covered in round } 0] \tag{4.83}$$

$$\leq \sum_{t=2}^{\infty} 2t \left(1 - 1\Delta^{\frac{1}{k}}\right)^{t-1} \tag{4.84}$$

$$= 2\frac{1}{1 - \left(1 - 1\Delta^{\frac{1}{k}}\right)} = 2\Delta^{\frac{1}{k}} \tag{4.85}$$

The law of total expectation yields that:

$$E\left[\mu(u)\right] \leq E\left[\mu(u) \mid u \textit{ covered in round } 0\right] + E\left[\mu(u) \mid u \textit{ not covered in round } 0\right] \tag{4.86}$$

$$= 3\Delta^{\frac{1}{k}}. \tag{4.87}$$

This proves the lemma.

Proof of Lemma 4.19

In the following, the authors fix an element $u \in V_{\mathscr{U}}$ and suppose it is covered in phase i. As one will see, all rounds of phase i have the same upper bound. For simplicity, do not condition on the specific round (of phase i) in which u is covered. The authors begin our proof by defining two *bad* events:

1. First, let $\mathscr{B}_1$ be the event that there is any set with span bigger than $6 \cdot \log \Delta \cdot \Delta^{\frac{1}{k}} \cdot \Delta_i$.

2. Second, let $\mathscr{B}_2$ be the event any set smaller than $\frac{\Delta_i}{4 \cdot \Delta^{\frac{1}{k}}}$ joins the solution.

The goal in phase i is to add sets that cover roughly Δ_i uncovered elements. So these two events imply that there are sets that greatly deviate from this value. The event $\mathscr{B}_1$ implies that there is a set with too many uncovered neighbors that should have been added earlier, while $\mathscr{B}_2$ implies that there is a set added *too early* as only a few uncovered neighbors for this phase. The worst case is, that both these events hold, and the difference between the sets of the largest and lowest span is huge. This assigns a high cost $\eta_{(i,j)}(u)$ to any element $u \in V_U$ that is covered. On the other hand, if neither of the two events holds, the authors bound $\eta_{(i,j)}(u)$ to

$$E[\eta_{(i,j)}(u) \mid \neg\mathscr{B}] \leq \frac{6 \cdot \log \Delta \cdot \Delta^{\frac{1}{k}} \cdot \Delta_i}{\Delta_i 4 \cdot \Delta^{\frac{1}{k}}} = 24 \cdot \log \Delta \cdot \left(\Delta^{\frac{1}{k}}\right)^2. \tag{4.88}$$

Hence, show that $\eta_{(i,j)}(u)$ is small in expectation. To do so, define event $\mathscr{B} := \mathscr{B}_1 \cup \mathscr{B}_2$ where both $\mathscr{B}_1$ and $\mathscr{B}_2$ hold and bound its probability. If one can show that $Pr[\mathscr{B}]$ is small, the lemma follows. In particular, assuming that $Pr[\mathscr{B}] \leq \frac{2}{\Delta}$, the law of total expectation implies:

$$E[\eta_{(i,j)}(u)c_{min}^{(i,j)}(u)] = Pr[\neg\mathscr{B}] \cdot E[\eta_{(i,j)}(u)c_{min}^{(i,j)}(u) \mid \neg\mathscr{B}] + Pr[\mathscr{B}] \cdot E[\eta_{(i,j)}(u)c_{min}^{(i,j)}(u) \mid \mathscr{B}] \tag{4.89}$$

$$\leq 24 \cdot \log \Delta \cdot \left(\Delta^{\frac{1}{k}}\right)^2 E[c_{min}^{(i,j)}(u) \mid \neg\mathscr{B}] + \frac{2}{\Delta} \cdot \Delta \cdot E[c_{min}^{(i,j)}(u) \mid \mathscr{B}] \tag{4.90}$$

$$\leq 25 \cdot \log \Delta \tag{4.91}$$

$$\cdot \left(\Delta^{\frac{1}{k}}\right)^2 \left(E[\eta_{(i,j)}(u)c_{min}^{(i,j)}(u) \mid \neg\mathscr{B}] + E[\eta_{(i,j)}(u)c_{min}^{(i,j)}(u) \mid \mathscr{B}]\right) \tag{4.92}$$

$$\leq 25 \cdot \log \Delta \cdot \left(\Delta^{\frac{1}{k}}\right)^2 E[c_{min}^{(i,j)}(u)] \tag{4.93}$$

In the following, the authors showed that $Pr[\mathscr{B}]$ is at most $\frac{2}{\Delta}$. For this, we need some further definitions. Let $H_u^i \subseteq N_u$ denote all neighbors that span more than $6 \cdot \log \Delta \cdot \Delta^{\frac{1}{k}} \cdot \Delta_i$ uncovered elements at any point in phase i and likewise let $L_u^i \subseteq N_u$ denote all neighbors that span less than $\frac{\Delta_i}{4 \cdot \Delta^{\frac{1}{k}}}$ at any point in the phase. It holds

$$Pr[\mathscr{B}_1] = Pr[\exists s \in H_u^i] \tag{4.94}$$

Furthermore, let S_i denote the sets that add themselves to the solution in phase i. Then, we obtain:

$$Pr[\mathscr{B}_2] = Pr[\bigcup_{s \in L_u^i} s \in S_i]. \tag{4.95}$$

Then, we have with the use of the union bound that:

$$Pr[\mathscr{B}] := Pr[\mathscr{B}_1 \cup \mathscr{B}_2] \tag{4.96}$$
$$\leq Pr[\mathscr{B}_1] + Pr[\mathscr{B}_2] \tag{4.97}$$
$$\leq Pr[\exists s \in H_u^i] + Pr[\bigcup_{s \in L_u^i} s \in S_i] \tag{4.98}$$

First, show that H_u^i is empty with prob. 2Δ. For this, consider the sets in H_u^i in the previous phase $i-1$. Define $\tilde{H}_u^{i-1}$ as the sets which had

$$6 \cdot \log \Delta \cdot \Delta^{\frac{1}{k}} \cdot \Delta_i := 6 \cdot \log \Delta \cdot \Delta_{i-1} \tag{4.99}$$

uncovered neighbors when they woke up in phase $i-1$. For these sets, it holds: Let $s \in V$ be a set and $c > 6$ be a constant. Assume that s has at least $c \cdot \log \Delta \cdot \Delta_{i-1}$ uncovered neighbors when it wakes up in phase $i-1$. Then, the probability that s is not added to the solution is smaller than $\Delta^{-(c-2)}$. Formally, we have:

$$Pr[s \notin S_{i-1} \mid s \in \tilde{H}_u^{i-1}] \leq \frac{1}{\Delta^{(c-2)}} \tag{4.100}$$

First, note that for $k < \log \Delta$, it holds $c \cdot \log \Delta \cdot \Delta_{i-1} > 4k$ as $c > 6$ is assumed. Now consider the round of phase $i-1$ where s woke up. The probability that in a fixed slot (of the $4k$ slots of that round), no elements beeps is at most:

$$Pr[\textit{No } \textsc{Beep} \textit{ in slot } \ell] \leq \left(1 - \frac{1}{\Delta_{i-1}}\right)^{c \cdot \log \Delta \cdot \Delta_{i-1}} \tag{4.101}$$
$$\leq e^{-c \log \Delta} = \frac{1}{\Delta^c} \tag{4.102}$$

This follows from the fact that s has $c \cdot \log \Delta \cdot \Delta_{i-1}$ uncovered neighbors. Recall that we initially assumed w.l.o.g. that Δ is bigger than $4k$. If $c \geq 6$, a union bound over all k slots then yields that all that with probability higher than $1 - \frac{1}{\Delta^2}$, in every slot, there was at least one active element in the neighborhood of s that beeped. In particular, s counted a Beep in more than $3k$ slots when it woke up. Thus, the set in question must have added itself to the solution in phase $i-1$. Formally, we have:

$$Pr[s \in S_{i-1}] \leq \sum_{\ell=1}^{4k} Pr[\textit{No } \textsc{Beep} \textit{ in slot } \ell] \tag{4.103}$$
$$\leq \sum_{\ell=1}^{4k} \frac{1}{\Delta^c} \leq \frac{1}{\Delta^{c-1}} \tag{4.104}$$

This was to be shown. Any set $s \in H_u^i$ must not have joined the solution in phase $i-1$. Suppose that a set has $c \cdot \log \Delta \cdot \Delta^{\frac{1}{k}} \cdot \Delta_i$ uncovered neighbors in phase i. Then it had at least $c \cdot \log \Delta \cdot \Delta_{i-1}$ uncovered neighbors when it woke up in phase $i-1$ as

their number can only decrease from phase to phase. So $s \in H_u^i$ implies $s \in \tilde{H}_u^{i-1}$. With the claim from above, the probability that s did not join the solution is

$$Pr[s \notin S_{i-1} \mid s \in \tilde{H}_u^{i-1}] \leq \frac{1}{\Delta^{6-4}} \leq \frac{1}{\Delta^2} \tag{4.105}$$

The claim with $c = 6$ is used, which follows from the definition of H_u^i. Hence, we obtain the desired bound on the sets that span many uncovered elements. Consider a set $s \in L_u^i$ and show that this set is unlikely to add itself to the solution in phase i. This formalizes the following claim. Let s be set with less than $\frac{\Delta_i}{4 \cdot \Delta^{\frac{1}{k}}}$ uncovered elements when it wakes up in phase i. Then, the probability that s joins the solution in phase i is lower than $\frac{1}{\Delta^2}$. More formally, let S_i denote that add themselves to the solution in phase i, then we have:

$$Pr[s \in S_i \mid s \in L_u^i] \leq \frac{1}{\Delta^2} \tag{4.106}$$

s will eventually wake up in some round phase i but it only adds itself to the solution if it hears a Beep in at least $3k$ slots of that round. Obviously, this requires at least $3k$ variables Y_u^ℓ (where u is an uncovered neighbor of s and ℓ is a slot) must be 1. We define:

$$\mathscr{B}_s := \sum_{u \in N_s, C_u = 0} \sum_{\ell=1}^{4k} Y_u^\ell \tag{4.107}$$

Thus, $\mathscr{B}_s$ is the sum of $4k \cdot \frac{\Delta_i}{4\Delta^{\frac{1}{k}}} < k\Delta$ variables as the have at most $\frac{\Delta_i}{4\Delta^{\frac{1}{k}}}$ uncovered elements and $4k$ rounds. Hence, the probability of B_s being bigger than $3k$ is

bounded as follows:

$$Pr[\mathscr{B}_s \geq 3k\ ] \leq \sum_{l=3k}^{k\Delta} \binom{4k \cdot \frac{\Delta_i}{4\Delta^{\frac{1}{k}}}}{l} \left(\frac{1}{\Delta_i}\right)^l = \sum_{l=3k}^{k\Delta} \binom{k \cdot \frac{\Delta_i}{\Delta^{\frac{1}{k}}}}{l} \left(\frac{1}{\Delta_i}\right)^l \tag{4.108}$$

$$\leq \sum_{l=3k}^{k\Delta} \left(\frac{e \cdot k \cdot \Delta_i}{\Delta^{\frac{1}{k}} \cdot l}\right)^l \frac{1}{\Delta_i^l} \tag{4.109}$$

$$\triangleright\ As\ \binom{n}{x} < (enx)^x \tag{4.110}$$

$$= \sum_{l=3k}^{k\Delta} \left(\frac{ek}{l}\right)^l \left(\frac{\Delta_i}{\Delta^{\frac{1}{k}}}\right)^l \frac{1}{\Delta_i^l} \leq \sum_{l=3k}^{k\Delta} \left(\frac{ek}{3k}\right)^l \left(\frac{\Delta_i}{\Delta^{\frac{1}{k}}}\right)^l \frac{1}{\Delta_i^l} \leq \sum_{l=3k}^{k\Delta} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^l \tag{4.111}$$

$$\triangleright\ As\ l \geq 3k \geq ek \tag{4.112}$$

$$= \sum_{l'=1}^{k\Delta-3k} \left(\frac{1}{\Delta^3}\right)^{l'} = \left(\frac{1}{\Delta^3}\right) \cdot \sum_{l'=1}^{k\Delta-3k} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{l'-1} \tag{4.113}$$

$$\triangleright\ Set\ l = 3k + (l' - 1) \tag{4.114}$$

$$\leq \left(\frac{1}{\Delta^3}\right) \cdot \sum_{l'=1}^{\infty} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{l'-1} \leq \left(\frac{1}{\Delta^3}\right) \cdot \sum_{l'=0}^{\infty} \left(\frac{1}{\Delta^{\frac{1}{k}}}\right)^{l'} \leq \frac{2}{\Delta^3} \tag{4.115}$$

$$\triangleright\ As\ \sum_{l'=0}^{\infty} a^{l'} \leq \frac{1}{1-a} \tag{4.116}$$

$$\leq \frac{1}{\Delta^2} \tag{4.117}$$

$$\triangleright\ As\ \Delta \geq 2 \tag{4.118}$$

Hence, the probability that s has beeping neighbors in $3k$ *distinct* slots and therefore adds itself to the solution, can only be smaller. This proves the claim.

Combine the two bounds. Then it holds:

$$Pr[\mathscr{B}] \leq Pr[\exists s \in H_u^i] + Pr[\bigcup_{s \in L_u^i} s \in S_i] \tag{4.119}$$

$$\leq Pr[\bigcup_{s \in H_u^i} s \notin S_{i-1} \mid s \in \tilde{H}_u^{i-1}] + Pr[\bigcup_{s \in L_u^i} s \in S_i] \tag{4.120}$$

$$\leq \Delta \frac{1}{\Delta^2} + \Delta \frac{1}{\Delta^2} = 2\Delta \frac{1}{\Delta^2} = \frac{2}{\Delta} \tag{4.121}$$

This concludes the proof.

5 Competitive Routing with Holes

In this chapter, we prove that the 2-localized Delaunay graph is a s-spanner and we present a routing strategy that finds c-competitive paths between any pair of nodes of the 2-localized Delaunay graph, in presence of holes but not yet with hole abstractions. The figures of the Delaunay triangulation were designed by Alina Ergova. The results are published in [24].

In general, we cannot apply routing strategies for the Delaunay graph in 2-localized Delaunay graphs because the 2-localized Delaunay graphs can contain holes. In this chapter, we prove that 2-localized Delaunay graphs and Delaunay graphs are very similar graphs in dense regions and that we can apply routing strategies for the Delaunay graph between visible nodes. Visible nodes are pairs of nodes whose direct line segment does not intersect any hole. For concrete definitions see section 5.1.

Theorem 5.1. *Let $G_{2Del} = (V, E_{2Del})$ be a 2-localized Delaunay graph and $s, t \in V$ such that the line segment $\overline{st}$ does not intersect any hole of G_{2Del}. Then, there exists a path p between s and t in G_{2Del} such that*

$$p \leq 1.998 \cdot ||st||.$$

Proof sketch: Let s and t be nodes of a Delaunay graph. Bose et al. consider the chain of triangles that is intersected by the line segment $\overline{st}$. Each of these triangles has an edge which either is above or below $\overline{st}$. We use only these edges and find out that these edges form a polygon. Going along all edges that lie above $\overline{st}$ is a path between s and t. We call this path the *upper chain* of s and t ($\mathrm{UC}(s,t)$) and we call the corresponding path for all edges that are below $\overline{st}$ the *lower chain* of s and t ($\mathrm{LC}(s,t)$). Xia proved that there is a path between any pair of nodes s and t in a Delaunay graph with path length at most $1.998 \cdot ||st||$. The path that is constructed by Xia contains only edges which connect nodes of $\mathrm{UC}(s,t)$ and $\mathrm{LC}(s,t)$. In Delaunay graphs a polygon that is described by an upper and a lower chain of nodes s and t does not contain any edge with a length larger than 1 under the assumption that s and t are visible from each other in the corresponding 2-localized Delaunay graph. Between any pair of visible nodes s and t in a 2-localized Delaunay graph there is a path with length at most $1.998 \cdot ||st||$. □

5.1 2-Localized Delaunay Graphs as Ad Hoc Networks with Holes

The Delaunay graph does not restrict the length of local edges, i.e., Wi-Fi edges. Hence, this graph structure is not useful to represent local edges in the ad hoc network as then edges may exceed the transmission range of a node.

The 2-localized Delaunay graph only allows edges which do not exceed the transmission range of a node and it is defined as follows

Definition 5.2. A triangle $\triangle(u, v, w)$ satisfies the 2-*localized Delaunay property* if

1. all edges of $\triangle(u, v, w)$ have length at most 1.
2. the interior of the disk $\bigcirc(u, v, w)$ does not contain any node which can be reached within 2 hops from u, v or w in UDG(V).

Definition 5.3. The $LDel^2(V)$ consists of

1. all edges of 2-localized triangles
2. all edges (u, v) for which the circle with diameter $\overline{uv}$ does not contain any further node $w \in V$ (*Gabriel Edges*)

In the case that the source and the target node of the 2-localized Delaunay graph are *visible* from each other, which means that their direct line segment does not intersect any hole, MixedChordArc is directly applicable. In the following figure we see this path between two nodes that are visible to each other.

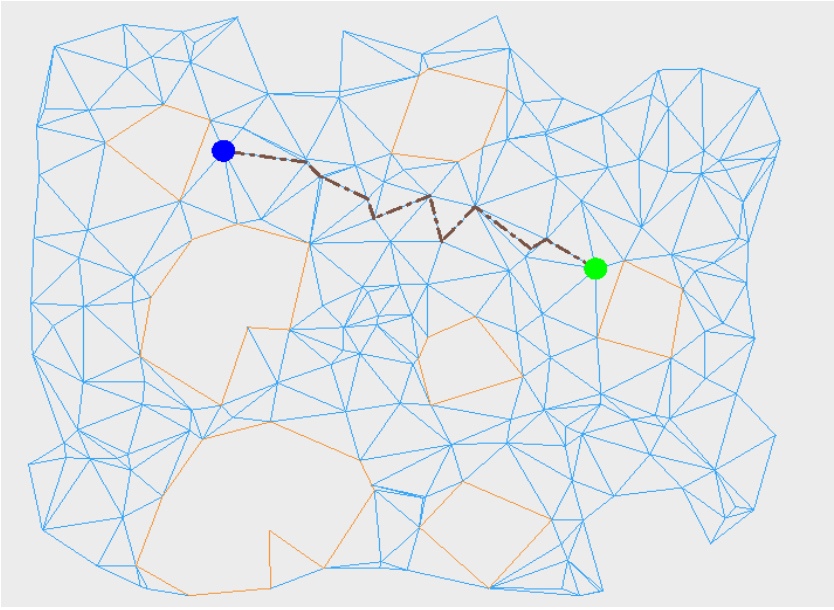

Figure 5.1: Mixed Chord Arc when source and target are visible from each other.

We do not want to store large routing tables. So our focus is on online routing strategies for this variant of the Delaunay graph. We use the visibility graph of holes to find *MixedChordArc*-competitive paths between any pair of nodes (s, t) by applying the *MixedChordArc*-strategy along every edge of the shortest path between s and t in the visibility graph. The visibility graph is defined as follows

Definition 5.4. In the visibility graph $Vis(V)$ of a set of polygons, V represents the set of corners of the polygons, and there is an edge $\{v, w\} \in Vis(V)$ if and only if a line can be drawn from v to w without crossing any polygon, i.e., v is visible from w.

and to formalize the length of the routing path, we have

Theorem 5.5. *Let s and t be two visible nodes of a 2-localized Delaunay graph. The Algorithm MixedChordArc finds a path between s and t with length at most* $3.56\|st\|$.

The 2-*localized* Delaunay graph does not contain all edges of a corresponding Delaunay graph. Thus, we cannot always use routing strategies for Delaunay graphs in our scenario, especially not in the presence of radio holes. We denote faces of the 2-localized Delaunay graph that are not triangles as *holes*.

Definition 5.6 (Hole)**.** Let $V \in \mathbb{R}^2$. A *hole* is a ring of nodes of the $LDel^2(V)$ with at least 4 nodes.

The following figure shows the restricted Delaunay graph, which is a Delaunay graph without edges that are larger than a fixed distance. The holes are drawn in orange.

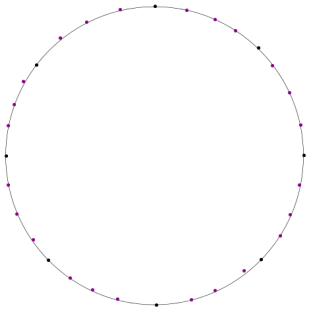

Figure 5.3: The hole nodes of the same hole form a ring.

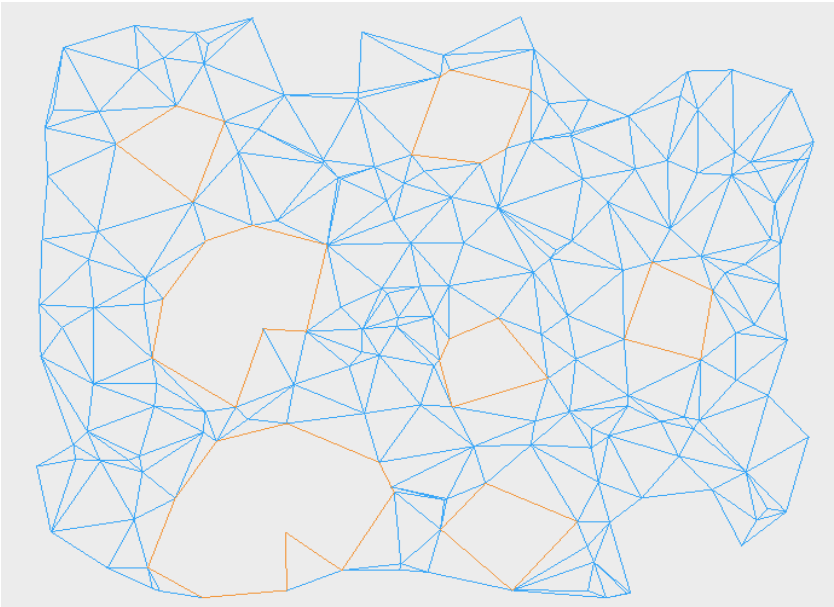

Figure 5.2: Restricted Delaunay graph.

The nodes that are on the perimeter of a hole are called *hole nodes*. It is easy to see, that the hole nodes of the same hole form a ring, i.e., each hole node is adjacent to exactly two other hole nodes for each hole, see the following figure

The choice of the 2-localized Delaunay graph as network topology is motivated by its *spanner*-property. We need this property to ensure the existence of competitive path in $LDel^2(V)$'s. Note that these graphs are no spanners of the Euclidean metric but they are spanners of the unit disk graph with 1.998 as the competitive-factor [41].

Theorem 5.7. *In local Delaunay graphs for $V \subset \mathbb{R}^2$, there exists a path between any pair of nodes s and t with length at most* 1.998 *times their distance in $UDG(V)$.*

We prove that 2-localized Delaunay graphs and Delaunay graphs are similar in

dense regions and hence we can apply routing strategies for the Delaunay graph between visible nodes.

Theorem 5.8. *Let $G_{2Del} = (V, E_{2Del})$ be a 2-localized Delaunay graph and $s, t \in V$ such that the line segment $\overline{st}$ does not intersect any hole of G_{2Del}. Then, there exists a path p between s and t in G_{2Del} such that*

$$p \leq 1.998 \cdot ||st||.$$

We prove Theorem 5.8 as follows: We use the technique that was introduced by Bose et al.

Lemma 5.9. Given a $G_{2Del} = (V, E_{2Del})$ and two nodes s and t such that the line segment $\overline{st}$ does not intersect any hole of G_{2Del}. Let $G_{Del} = (V, E_{Del})$ be the Delaunay graph to the same point set V. The polygon described by $\mathrm{UC}(s,t)$ and $\mathrm{LC}(s,t)$ in G_{Del} does not contain any edge e with $||e|| > 1$.

Proof: We consider three different types of edges which are part of the polygon with boundaries $\mathrm{UC}(s,t)$ and $\mathrm{LC}(s,t)$ in G_{Del}. The edges are edges that cross the line segment $\overline{st}$, edges that lie completely above $\overline{st}$ and edges that lie completely below $\overline{st}$. We prove for every different type of these edges that they cannot they are at most one in case s and t are visible from each other in G_{2Del}.
Case 1: Edges crossing $\overline{st}$:
Let $\triangle abc$ be a triangle which is intersected by $\overline{st}$ and let $\overline{ab}$ be an edge that crosses $\overline{st}$. We assume that $||ab|| > 1$. This immediately implies that $\overline{st}$ crosses a hole because $\overline{st}$ intersects a face with at least 4 nodes which gives a contradiction to our assumption.
Case 2: Edges above $\overline{st}$:
Let $\triangle abc$ be a triangle which is intersected by $\overline{st}$ and let $\overline{ab}$ be an edge that lies above $\overline{st}$ with $||ab|| > 1$. With the knowledge above we conclude that $\overline{ac}$ and $\overline{bc}$ are on the perimeter of a hole if $\overline{ac}$ and $\overline{bc}$ are not hole edges themselves. We can easily see that $\overline{st}$ would cross a hole in this case and is a contradiction to our assumption.
Case 3: Edges below $\overline{st}$:
We apply the same argumentation as for Case 2 in this case.
Because every possible type of edges is not greater than 1, we have proven this lemma. □

Lemma 5.9 implies that we can apply the same routing strategies for Delaunay Graphs also between visible nodes in 2-localized Delaunay graphs.

De Berg et al. [11] proved that in polygons, the shortest path between two nodes in the plane contains obstacle nodes. So, if we consider the Visibility Graph of holes of the 2-localized Delaunay graph, we translate a path in the visibility graph to a path in 2-localized Delaunay graphs by applying a routing strategy for Delaunay graphs along every edge on the path in the visibility graph.

5.2 Routing Protocol

In this section, we use the hybrid communication model to introduce a routing strategy for 2-localized Delaunay graphs. We use long-range links of the cellular infrastructure to exchange control information about locations and shapes of holes. The distributed calculations for this are presented in Chapter 9. We assume the knowledge about the shape of holes and focus on the routing strategy itself. We also reduce the storage requirements. We will improve the routing strategy in Section 6 with respect to the storage requirements and the distributed computation time. More details about establishing a 2-localized Delaunay graph of ad hoc links provides Section 9.3.

Throughout this section, we assume a correct 2-localized Delaunay graph. We assume that every node that is located on the perimeter of a hole stores a visibility graph of all hole nodes. Later on in Section 9.1, we consider the aggregating the information that is needed for the visibility graphs.

The direct lines will give us the direction for routing on competitive paths for the 2-localized Delaunay graph even when network holes are present.

The routing protocol *VisibilityHole* routes messages between two nodes s and t in the 2-localized Delaunay graph $LDel^2(V)$, when hole nodes store the visibility graph to obtain routing directions as follows: A source s that wants to send a message to a target node t, initially contacts t via a long-range link to ask for $t's$ geographical position which is a tuple of coordinates (t_x, t_y). t replies with its position via a direct long-range link and then s sends its message via MixedChordArc towards (t_x, t_y). We distinguish the two cases:

1. The message reaches t via *MixedChordArc*.

2. The message reaches a hole node h_0, i.e., the direct line segment $\overline{st}$ intersects a hole. Then, h_0 inserts t into its visibility graph and applies a shortest path algorithm on edges of the visibility graph from itself to t. The resulting shortest path $(h_0, h_1, h_2, \ldots, h_k = t)$ is then used to transmit the message via ad hoc links. After reaching h_1, the procedure is repeated until the message finally reaches t.

The following figure gives an example when routing from the blue to the green node that are not visible to each other.

Theorem 5.10 (routingWithHole)**.** *The routing protocol* VisibilityHole *routes messages from source s to target t in $LDel^2(V)$ on competitive paths.*

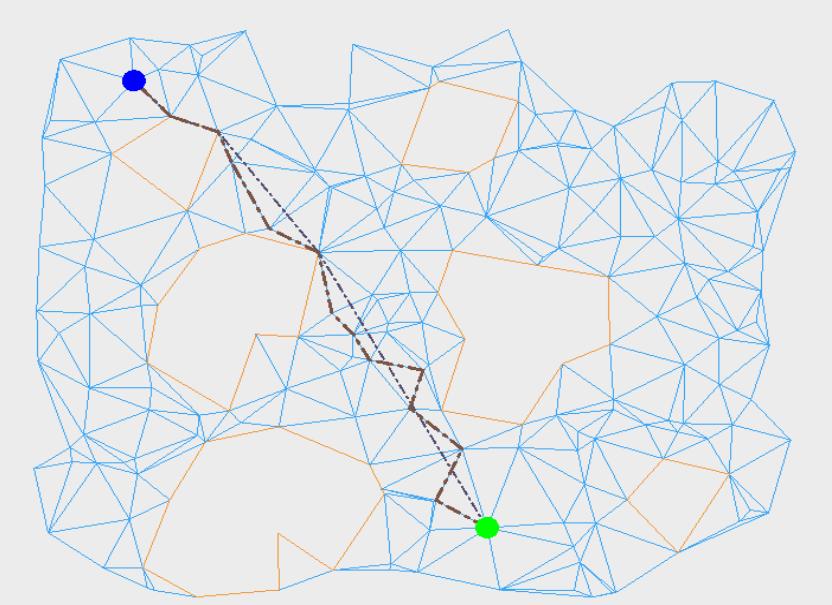

Figure 5.4: Visibility graph between hole nodes and MixedChordArc when source and target are not visible from each other.

Proof: In case (1), there is no hole under the direct line between s and t. Hence, by routing with MixedChordArc we have a 3.56 -competitive path from s to t.
In case (2), let p_{st} be the shortest path between s and t in the Visibility Graph. In case h_0 is on the shortest path between s and t in the Visibility Graph, the related path in the 2-localized Delaunay Graph has length at most $3,56 \cdot \|p_{st}\|$. Otherwise, the initial path to h_0 would be a detour. We see that the detour increases the competitiveness only by a constant factor. Because MixedChordArc did not reach t but instead it reaches a node h_0, it follows that the path that is taken from s to h_0 has length less or equal to $3.56 \cdot \|st\|$ and this is less or equal to 3.56 times the shortest possible path between s and t in the 2-localized Delaunay Graph. Thus, the detour increases the competitive constant only by a factor of 3. Finally, we obtain an 10.68-competitive path between s and t. □

Unfortunately, the perimeter nodes of radio holes can store a huge Visibility Graph. This number of nodes and edges they have to store is large. There could be even holes in 2-localized Delaunay graph with $\Theta(n)$ nodes on its perimeter. Also, let h denote the number of nodes on the perimeter of a hole, then the Visibility Graph may contain up to $\Theta(h^2)$ edges.

To reduce the number of edges to $\mathcal{O}(h)$ we do not compute the entire visibility graph but we compute a Delaunay graph of all nodes lying on different holes. We call this the *overlay Delaunay graph of hole nodes*. The following figure provides an example.

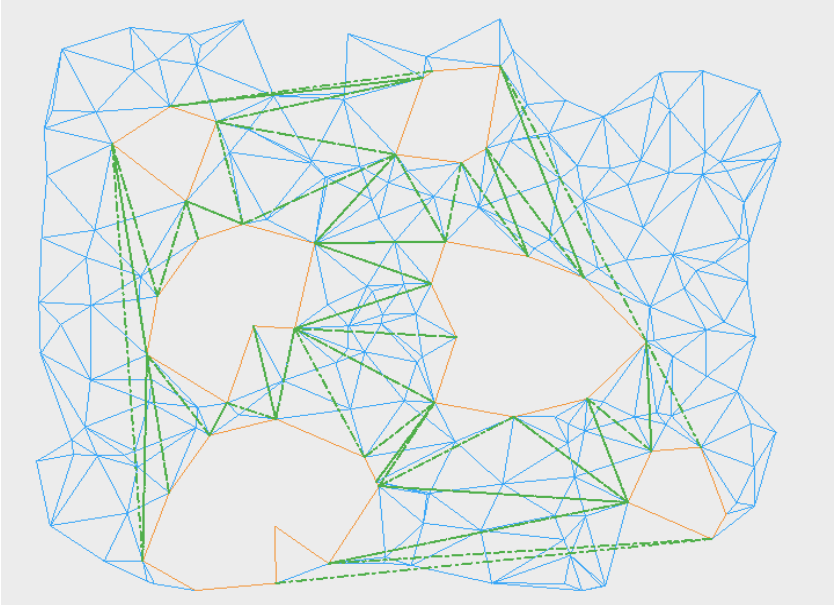

Figure 5.5: Delaunay triangulation between all hole nodes.

Finally, we obtain

Theorem 5.11 (ReductionHoles)**.** *The* overlay Delaunay graph of hole nodes *stores* $\mathcal{O}(h)$ *edges, where* h *is the number of holes.*

Proof: Because Delaunay Graphs are planar graphs, this reduces the number of edges to $\mathcal{O}(h)$. It also affects the obtained length of the paths. In general, Delaunay Graphs do not contain the shortest connection between two nodes but they contain a path which is 1.998-competitive. Thus, by using a Delaunay Graph instead of a Visibility Graph, we obtain a path length of $1.998 \cdot 10.68 \cdot \|p_{st}\| \leq 21.33864 \cdot \|p_{st}\|$. □

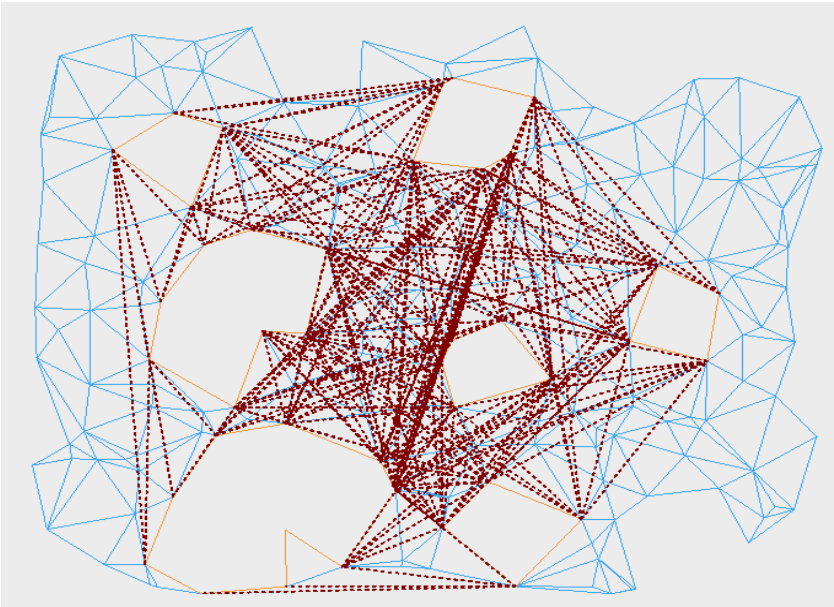

Figure 5.6: Visibility graph between all hole nodes.

6 Convex Hulls as Hole Abstraction

In Section 5.2, we mentioned the advantage of a visibility graph and a Delaunay graph of all hole nodes to find c-competitive paths in ad hoc networks. Nevertheless, the storage requirements for each single hole node are linear in the total number of hole nodes. But how can the number of nodes in the visibility graph be reduced even further while still computing competitive paths? In Section 6.1, we investigate that convex hull nodes of holes reduce the space requirements significantly. Throughout this chapter, we assume that the convex hulls of holes do not intersect, as illustrated in the following figure.

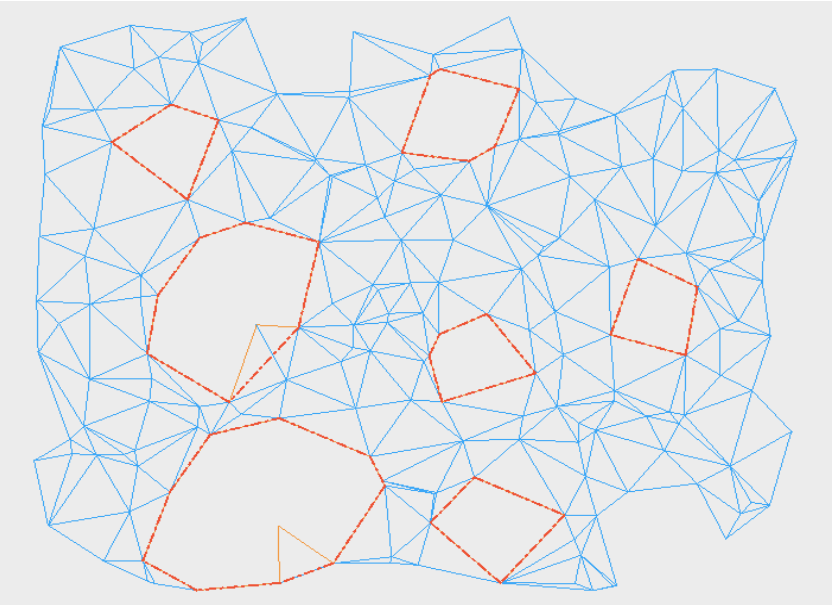

Figure 6.1: Restricted Delaunay graph with holes and convex hulls as hole abstraction.

We prove in Section 6.2 that convex hulls still help us to find competitive paths between almost all source-destination pairs in the ad hoc network. Based on these observations, we introduce in Section 6.3 a c-competitive routing strategy. This is similar to our protocol of Section 5.2 which considers only hole nodes that are part of the convex hulls of holes instead of all hole nodes. The figures of the Delaunay triangulation were designed by Alina Ergova. The results in this chapter are published in [24].

6.1 Space Reduction

To obtain a space reduction from the number of all hole nodes we first focus on locally convex hulls of the radio holes.

Definition 6.1. Let $(v_1, v_2, \ldots, v_k, v_1)$ be a cycle of nodes in the local Delaunay graph at the perimeter of some hole. We call $(v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}, v_{i_1})$ for some $1 \leq i_1 < i_2 < \ldots, i_\ell \leq k$ a *locally convex hull* of that hole if
(1) $\|v_{i_j} v_{i_{j+1}}\| \leq 1$ for all $j \in \{1, \ldots, \ell\}$ (where $v_{i_{\ell+1}} = v_{i_1}$), and
(2) there are no 3 consecutive nodes u, v, w in that sequence where $\angle(u, v, w) \geq 180°$ and $\|uw\| \leq 1$.

Concerning the number of nodes of locally convex hulls, we have

Lemma 6.2. For any cycle $(v_1, v_2, \ldots, v_k, v_1)$ of hole nodes in the local Delaunay graph that covers an area of size A, any locally convex hull of that cycle contains $\mathcal{O}(A)$ nodes.

Proof Consider a locally convex hull $(v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}, v_{i_1})$. Let u, v, w be 3 consecutive nodes in the just mentioned sequence. If $\angle(u, v, w) \geq 180°$, then it follows from the definition of the locally convex hull that $\|uw\| > 1$. If $\angle(u, v, w) < 180°$, then $\|uw\| > 1$ as well because otherwise v would not be on the perimeter of the hole. It follows for the predecessor p of u and the successor s of w that $\|pv\| > 1$ and $\|vs\| > 1$. Further, there cannot be any other node $x \in \{v_{i_1}, \ldots, v_{i_\ell}\}$ with $\|vx\| \leq 1$ because otherwise we would have a shortcut in the perimeter. This means that $(v_1, v_2, \ldots, v_k, v_1)$ cannot be the perimeter of a hole. Thus, the unit cycle around each v_{i_j} contains at most 2 other nodes of the locally convex hull. This implies $\ell = \mathcal{O}(A)$. □

Hence, a locally convex hull contains a number of nodes that is independent of the total number of nodes in the system. It only depends on the area that is covered by the hole. A further reduction in the number of nodes can be done by the convex hull of a hole.

Lemma 6.3. For any cycle $(v_1, v_2, \ldots, v_k, v_1)$ of hole nodes in local Delaunay graph with a bounding box (i.e., the box of minimum size containing $v_1, \ldots, v_k$) of circumference L, the convex hull $(v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}, v_{i_1})$ of the cycle contains $\mathcal{O}(L)$ nodes.

Proof Let B be the bounding box of the cycle and let x be its center point. Let the points $w_{i_1}, \ldots, w_{e_\ell}$ be the projections of $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ from x onto the boundary of B. These are the points where the ray from x in the direction of v_{i_j} intersects the boundary of B. It is easy to see that the ℓ_1-distance of w_{i_j} and $w_{i_{j+1}}$ on B is at least as large as $\|v_{i_j} v_{i_{j+1}}\|$ for all j. Also, for any 3 consecutive points u, v, w on the convex hull it holds that $\|uw\| > 1$. Thus, for any 3 consecutive points u', v', w' on the projection of the convex hull onto B we have that the ℓ_1-distance of u' and w' is greater than 1. This implies that the convex hull contains only $\mathcal{O}(L)$ nodes. □

A bounding box of circumference L can cover an area of size $\Theta\left(L^2\right)$. Hence, we get another significant reduction in the number of nodes when we consider, instead of locally convex hulls, the convex hulls. If we consider convex hulls of holes, we achieve a significant reduction of the number of nodes that are contained in the visibility graph.

6.2 c-competitive Paths via Convex Hulls

In this section, we prove that considering nodes of convex hulls of holes still allows us to find competitive paths in the 2-localized Delaunay graph. We assume that both the source and the target of a routing request are outside of any convex hull. In addition, we assume that the source and the target are not visible from each other because finding c-competitive paths for visible nodes can be found via *MixedChordArc*.

Lemma 6.4. The shortest path between any pair of non-visible nodes of the 2-localized Delaunay graph contains convex hull nodes.

Proof The basic idea is that, if a node is part of the interior of a convex hull and it is part of the shortest path, then there exists a shorter path. This is a contradiction. In more detail, let s, t be two nodes of the 2-localized Delaunay graph, whose direct line segment intersects a hole. When starting from s, we consider ℓ to be the first intersected line segment of the boundary of the intersected convex hull with the endpoints v, w. Assume that the shortest path contains the points of $(v, ..., w)$. Else, the argumentation hast to be repeated with the neighboring edges of the convex hull. By contradiction, assume that the shortest path from s to t contains a point $p \in (v, ..., w)$ from the interior of the convex hull, i.e., excluding v, w. We assume that the shortest path contains also the point w. We assume that the same holds for v. Because of the triangle inequality, we have: $\|sw\| \leq \|sp\| + \|pw\|$. We also know that $\| (x, y) \| \leq 1.998 \cdot \|xy\|$ holds for any two nodes of a Delaunay triangulation.
Then:

$$\begin{aligned}\frac{\| (s, w) \|}{1.998} &\leq \|sw\| \\ &\leq \|sp\| + \|pw\| \\ &\leq \| (s, p) \| + \| (p, w) \| \\ &= \| (s, ..., p, ..., w) \|\end{aligned}$$

Thus, the points of the interior of convex hulls are not chosen because a path along a convex hull node would be shorter. □

Using the lemma, we show that a Delaunay graph of all convex hull nodes allows us to find competitive paths in the 2-localized Delaunay graph. We consider the

overlay Delaunay graph to be a Delaunay graph that contains all convex hulls of holes and that connects the nodes of different convex hulls in a Delaunay graph, see the following figure as a visualization.

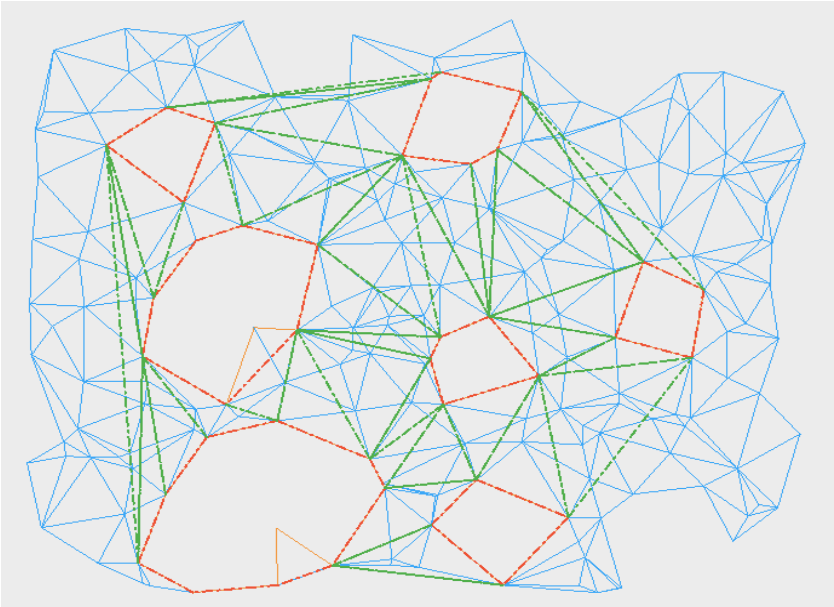

Figure 6.2: Overlay Delaunay graph between convex hull nodes.

The following theorem is a conclusion of the properties mentioned above.

Theorem 6.5. *Let s and t be two nodes of a 2-localized Delaunay graph that do not lie inside of any convex hull. Further, let $(s = c_0, c_1, \ldots, c_{\ell-1}, c_\ell = t)$ be the shortest path in the overlay Delaunay graph via long-range links. Then we have*

1. *There is a $\left(1.998 \cdot \sum_{m=0}^{\ell-1} d_m\right)$-path in the 2-localized Delaunay graph from s to t, where $d_m := \|c_m c_{m+1}\|$.*

2. *By applying* MixedChordArc*algorithm, we obtain a $\left(3.56 \cdot \sum_{m=0}^{\ell-1} d_m\right)$-path in the 2-localized Delaunay Graph from s to t, where $d_m := \|c_m c_{m+1}\|$.*

See also the following figure for a better visual understanding.

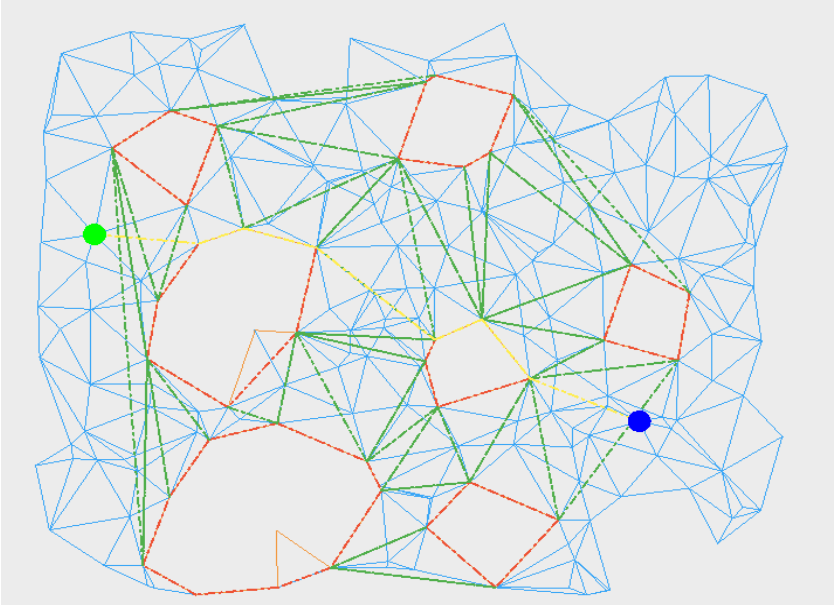

Figure 6.3: Shortest path overlay (yellow) between source node (green) and target node (blue).

This approach finds c-competitive paths from source s to target t in the 2-localized Delaunay graph. We will prove Theorem 6.5(1) with the following two lemmata:

Lemma 6.6. Let a and b be visible nodes of different convex hulls. Then there is a $1.998 \cdot \|ab\|$-spanning path between them in the 2-localized Delaunay graph.

Proof Here, we use the observation of Xia that a 2-localized Delaunay graph is a 1.998-spanner of the unit disk graph and Lemma 5.11. Because the Delaunay graph is a 1.998-spanner of the complete Euclidean graph and because the 2-localized Delaunay graph contains all edges of the original Delaunay graph between the upper and lower chain of a pair of visible nodes, we know that there always exists a $1.998 \cdot \|ab\|$ path between two visible nodes a and b of two different convex hulls. □

Lemma 6.7. Let a and b be adjacent nodes on a convex hull, where $a \neq b$. Then there is a $1.998 \cdot \|ab\|$-spanning path in the 2-localized Delaunay graph between a and b.

Proof We know by Xia that a 2-localized Delaunay graph is a 1.998-spanner of the unit disk graph. Hence, there is a 1.998-competitive path between the two convex hull nodes. □

To prove Theorem 6.5(2), we consider the following lemmata:

Lemma 6.8. Let a and b be visible nodes of different convex hulls. Then there is a $3.56 \cdot \|ab\|$-routing path between them in the 2-localized Delaunay graph.

Proof This lemma follows from the fact that for two visible nodes s and t, their direct line segment $\overline{st}$ intersects only triangles which are part of the Delaunay graph. □

Lemma 6.9. Let a and b be adjacent nodes on a convex hull, where $a \neq b$. Then there is a $3.56 \cdot \|ab\|$-routing path in the 2-localized Delaunay graph between a and b.

Proof The proof is similar to the proof of Lemma 6.8. Two adjacent convex hull nodes are by the assumption of non-intersecting convex hulls per definition visible from each other. □

7 Bounding Boxes as Hole Abstractions

In the previous chapters, we reduced the number of hole nodes to the number of convex hulls as hole abstractions. We can even reduce the number of convex hull nodes to the number of bounding box nodes. Therefore, we consider bounding boxes as hole abstractions. The results of this chapter were published in [6] and [5].

7.1 Bounding Boxes and Virtual Axis

Because bounding box nodes of holes do not always match with nodes of the ad hoc network, see Figure 7.2), we consider an embedding of bounding boxes in the 2-localized Delaunay graph. For a 2-localized Delaunay graph with node set V and edge set E, we are interested in *representatives* of a bounding boxes in V.

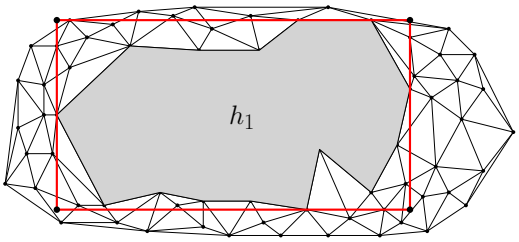

Figure 7.1: The nodes of the red bounding box are not part of the 2-Localized Delaunay graph.

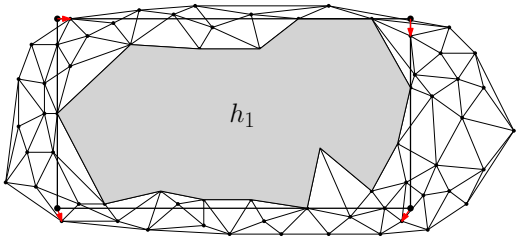

Figure 7.2: A bounding box described by its representatives of the ad hoc network.

Hence, we consider the nodes of V that have the shortest Euclidean distance to a bounding box node. Let $G_{2Del} = (V, E)$ be a two localized Delaunay graph with the corresponding Voronoi diagram $\mathrm{Vor}(V)$. A bounding box node b is represented by the node of the Voronoi cell $c \in \mathrm{Vor}(V)$ with $b \in c$. The obtained bounding box, see Figure 7.2, does not enclose the entire hole. We will prove it to have similar properties as the represented bounding box. We know about the existence of c-competitive paths between visible nodes in 2-localized Delaunay graphs. We use the direct line between bounding box nodes for routing. We call this direct line *virtual axis* and this is defined as follows.

Definition 7.1 (Virtual Axis)**.** Consider a $G_{2Del} = (V, E)$ with nodes $s, t \in V$. Let C_s and C_t be the cells of the corresponding Voronoi diagram with $s \in C_s$ and $t \in C_t$. Additionally let $a, b \in \mathbb{R}^2$ with $a \in C_s$ and $b \in C_t$ but $a, b \notin V$ and $a \neq b$. We call the line segment $\overline{ab}$ a *virtual axis* between s and t in G_{2Del}. For the ease of notation, we simply write $\mathrm{vAxis}(s, t)$.

The following theorem says that between every pair of representatives of adjacent bounding box nodes there exists a path in the 2-localized Delaunay graph with length at most 3.996 times the Euclidean distance between the bounding box nodes.

Theorem 7.2. *Let $G_{2Del} = (V, E)$ be a 2-Localized Delaunay graph with $s, t \in V$. For any* vAxis(s, t) *with endpoints $bb_{t\ell}$ and bb_{tr}, where s and t are representatives of $bb_{t\ell}$ and bb_{tr}, that does not intersect any hole of G_{2Del}, there exists a path p between s and t in G_{2Del} with length at most:*

$$||p|| \leq 3.996 \cdot ||bb_{t\ell}bb_{tr}||.$$

We prove Theorem 7.2 with two lemmas. Lemma 7.3 bounds the length of the line segment $\overline{st}$. Lemma 7.6 proves that vAxis(s, t) is the shortest polyline, see Definition 7.5, between s and t. The combination of both lemmas yields then to Theorem 7.2.

Lemma 7.3. Let $G_{2Del} = (V, E, w)$ be a 2-localized Delaunay graph with a pair of nodes $s, t \in V$ and let vAxis(s, t) be a virtual axis between s and t that does not intersect any hole of G_{2Del}. The endpoints of vAxis(s, t) are denoted as $bb_{t\ell}$ and bb_{tr}. Then,

$$||st|| \leq 2 \cdot ||bb_{t\ell}bb_{tr}||.$$

Proof By the triangle inequality $||st|| \leq ||\overline{sbb_{t\ell}}|| + ||bb_{t\ell}bb_{tr}|| + ||\overline{bb_{tr}t}||$. Let us bound the length of the line segments $\overline{sbb_{t\ell}}$ and $\overline{bb_{tr}t}$. The length of each line segment is at most $\frac{1}{2}$. s is the representative of $bb_{t\ell}$ because s is the node with the smallest distance to $bb_{t\ell}$ of all nodes in V. $bb_{t\ell}$ is either in or on the boundary of a triangle t_s that contains s as a node. Each edge of this triangle has length at most 1. It is easy to see that the endpoints of a triangle – t_s in our case – are those nodes with the largest distances to each other in the triangle. The worst case is that $bb_{t\ell}$ is exactly on the half of an edge with length 1. Then the closest point is at distance $\frac{1}{2}$ which is an upper bound for $||bb_{t\ell}s||$. We say the same for the line segment $\overline{bb_{tr}t}$. Let us bound the length of $\overline{st}$ as follows:

$$\begin{aligned} ||st|| &\leq ||sbb_{t\ell}|| + ||bb_{t\ell}bb_{tr}|| + ||bb_{tr}t|| \\ &\leq \frac{1}{2} + ||bb_{t\ell}bb_{tr}|| + \frac{1}{2} \\ &= ||bb_{t\ell}bb_{tr}|| + 1 \end{aligned}$$

Further, $||bb_{t\ell}bb_{tr}|| > 1$, due to the definition of holes. Thus, we obtain a final bound on $||st||$:

$$\begin{aligned} ||st|| &\leq ||bb_{t\ell}bb_{tr}|| + 1 \\ &\leq ||bb_{t\ell}bb_{tr}|| + ||bb_{t\ell}bb_{tr}|| \\ &= 2 \cdot ||bb_{t\ell}bb_{tr}||. \end{aligned}$$

□

After being able to express the length of $\overline{st}$ in terms of $||bb_{t\ell}bb_{tr}||$, we prove that there is a c-competitive path between s and t. The proof is based on the path construction for Delaunay graphs introduced by Xia. We use two definitions that have been introduced by Xia [41].

Definition 7.4 (Chain of Disks)**.** A finite sequence of disks $\mathscr{O} = (O_1, O_2, \ldots, O_n)$ is called *chain of disks* if it has the following two properties:

Property 1: Every pair of consecutive disks O_i and O_{i+1} intersects but neither disk contains the other. Denote by $C_i^{(i-1)}$ and $C_i^{(i+1)}$ the arcs on the boundary of O_i that are in O_{i-1} and O_{i+1}, respectively. These arcs are denoted as *connecting arcs* of O_i. **Property 2:** The connecting arcs of O_i do not overlap for $2 \leq i \leq n-1$.

However they can share an endpoint. Two points u and v are called *terminals* of $\mathscr{O}$ if u lies on the boundary of O_1 and is not in the interior of O_2 and v lies on the boundary of O_n and is not in the interior of O_{n-1}.

Definition 7.5 (Shortest polyline between u and v)**.** Given a chain of disks $\mathscr{O} = (O_1, O_2, \ldots, O_n)$ with terminals u and v, let $o_1, \ldots, o_n$ be the centers of $O_1, \ldots, O_n$. The polyline $uo_1 \ldots o_n v$ is called the *centered polyline* between u and v. For $1 \leq i \leq n-1$, let a_i and b_i be the intersections of the boundaries of O_i and O_{i+1}. Without loss of generality, all a_i's are assumed to be on one side of the centered polyline and all b_i's are on the other side. For notational convenience, define $a_0 = b_0 = u$ and $a_n = b_n = v$. Let $D_{\mathscr{O}}(u, v) = up_1 \ldots p_{n-1}v$ be the *shortest polyline* from u to v that consists of line segments $\overline{up_1}, \overline{p_1p_2}, \ldots \overline{p_{n-1}v}$ where $p_i \in \overline{a_ib_i}$ for $1 \leq i \leq n-1$.

With these definitions, we prove Lemma 7.6.

Lemma 7.6. Let $G_{2Del} = (V, E, w)$ be a 2-localized Delaunay graph with a pair of nodes $s, t \in V$ and let vAxis(s, t) be a virtual axis that does not intersect any hole of G_{2Del} with endpoints $bb_{t\ell}$ and bb_{tr}. Further let $\mathscr{O}$ be the chain of disks with terminals s and t obtained by the circumcircles of all triangles intersected by vAxis(s, t). Then, we can bound the shortest polyline $D_{\mathscr{O}}(s, t)$ as follows:

$$D_{\mathscr{O}}(s, t) \leq 2 \cdot ||bb_{t\ell}bb_{tr}||.$$

Proof By [41], any sequence of disks obtained by the circumcircles of triangles along a line segment in a Delaunay graph is a chain of disks. With Lemma 5.9, we know that vAxis$(s, t) = \overline{bb_{t\ell}bb_{tr}}$ does not intersect any holes. Use the chain of disks $\mathscr{O}$ with terminals s and t obtained by the circumcircles of all triangles intersected by vAxis(s, t). See Figure 7.3 as an example. The main argument is that the polyline $||sbb_{t\ell}bb_{tr}t||$ fulfills the requirements of the previous definition and it is a candidate for the shortest polyline between s and t. Like this, $||sbb_{t\ell}bb_{tr}t||$ is an upper bound

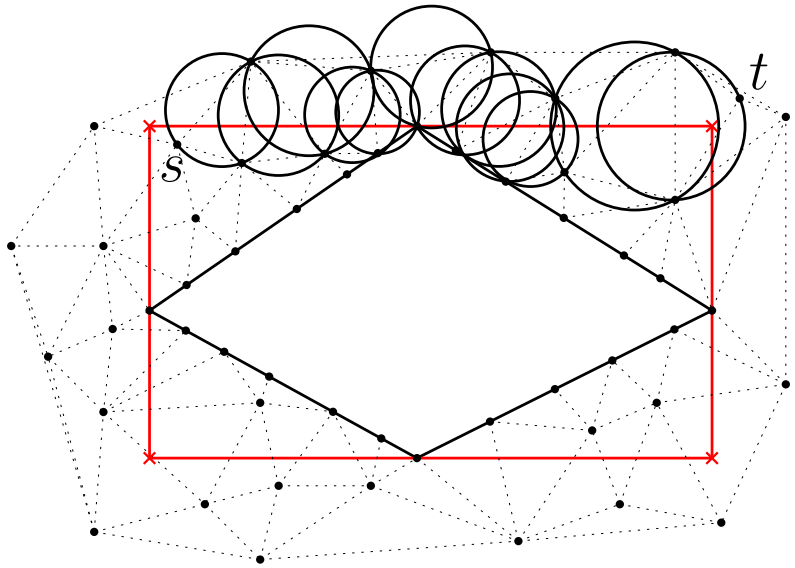

Figure 7.3: The chain of disks $\mathscr{O}$ from s to t along vAxis(s,t).

.

for $D_{\mathscr{O}}(s,t)$. So:

$$\begin{aligned} D_{\mathscr{O}}(s,t) &\leq ||sbb_{t_\ell}|| + ||bb_{t\ell}bb_{tr}|| + ||bb_{tr}t|| \\ &\overset{Lemma 7.3}{\leq} 2 \cdot ||bb_{t\ell}bb_{tr}|| \end{aligned}$$

□

Lemma 7.3 and Lemma 7.6 help to prove Theorem 7.2. Xia mentioned that the shortest connection $P_{\mathscr{O}}(s,t)$ between two terminal nodes s and t along a chain of disks is at most $1.998 \cdot D_{\mathscr{O}}(s,t)$ [41]. There is a path between s and t with length at most:

$$\begin{aligned} P_{\mathscr{O}}(s,t) &\leq 1.998 \cdot D_{\mathscr{O}}(s,t) \\ &\overset{Lemma 7.6}{\leq} 1.998 \cdot 2 \cdot ||bb_{t\ell}bb_{tr}|| = 3.996 \cdot ||bb_{t\ell}bb_{tr}||. \end{aligned}$$

So far, we proved the existence of this path. In addition, we find a c-competitive path via the MixedChordArc algorithm. We modify the algorithm and do not use the direct line segment between two representatives as the referencing segment. Instead we use the virtual axis connecting the real bounding box vertices. The entire path has a length of at most 5.56 times the length of the virtual axis. The reason for this is that the connection between s and the first node along the path and t and the last node on the path has a length at most 2 times the length of the virtual axis. This gives the following lemma.

Lemma 7.7. Let $G_{2Del} = (V, E)$ be a 2-localized Delaunay graph with $s, t \in V$. For any vAxis(s, t) with endpoints $bb_{t\ell}$ and bb_{tr} that does not intersect any hole of G_{2Del}, there exists an online routing strategy that finds a path p between s and t in G_{2Del} with length at most:

$$p \leq 5.56 \cdot ||bb_{t\ell} bb_{tr}||.$$

7.2 Competitive Paths

In this section, we prove the existence of c-competitive paths between bounding box nodes. We introduce *Bounding Box Visibility Graphs*, where each hole is represented by its axis-parallel bounding box. The node set V consists of the nodes of the axis-parallel bounding box of each hole. The edge set E consists of the edges of each bounding box as well as of edges between visible nodes of different bounding boxes. Let O be a set of polygons and let $s, t \in \mathbb{R}^2$. Further let $bb_{t\ell}(p), bb_{tr}(p), bb_{b\ell}(p)$ and let $bb_{br}(p)$ be the nodes of an axis-parallel bounding box that represent a polygon $p \in O$. The bounding box visibility graph is defined as follows:

Definition 7.8 (Bounding Box Visibility Graph)**.** A geometric graph $G = (V_{BB}, E)$ is called *bounding box visibility graph* if $bb_{t\ell}(p), bb_{tr}(p), bb_{b\ell}(p)$ and $bb_{br}(p) \in V_{BB}, \forall p \in O$. Additionally, $(bb_{t\ell}(p), bb_{tr}(p)), (bb_{t\ell}(p), bb_{b\ell}(p)), (bb_{tr}(p), bb_{br}(p))$ and $(bb_{b\ell}(p), bb_{br}(p)) \in E, \forall p \in O$.

For two nodes of different bounding boxes $u, v \in V_{BB}$, the edge $\{u, v\} \in E$ if $\overline{uv}$ does not intersect the bounding box of any obstacle $p \in O$.

The following figure gives an example of the bounding box visibility graph.
This graph provides a special geometrical property.

Theorem 7.9. *Let $G_{BB} = (V_{BB}, E)$ be a Bounding Box Visibility Graph of an ad hoc network that contains multiple non-intersecting bounding boxes and a source- and a target-location s and t, $s, t \in G_{BB}$. There exists a path p_{st}^{BB} between s and t with:*

$$p_{st}^{BB} \leq \sqrt{2} \cdot d_{\mathrm{UDG}}(s, t).$$

To prove the theorem above, we define a certain class of paths in geometric graphs. These graphs help us to construct paths in bounding box visibility graphs which are c-competitive to the shortest path in visibility graphs. We compare the covered distance in both the vertical and horizontal direction of both paths.

Definition 7.10 (x/y-monotone paths)**.** A path $p = (p_1, p_2, \ldots, p_k)$ in a geometric graph is called *increasing x-monotone* if $x(p_i) \leq x(p_{i+1})$ for all $i \in \{1, \ldots, k-1\}$. Analogously, such a path is called *increasing y-monotone* if $y(p_i) \leq y(p_{i+1})$ for all $i \in \{1, \ldots, k-1\}$. Similarly, paths are called *decreasing x-/y-monotone* if $x(p_i) \geq x(p_{i+1}) / y(p_i) \geq y(p_{i+1})$ for all $i \in 1, \ldots, k-1$. A path is called *x-monotone* if it is either increasing or decreasing x-monotone. Analogously, a path is called *y-monotone* if it is either increasing or decreasing y-monotone.

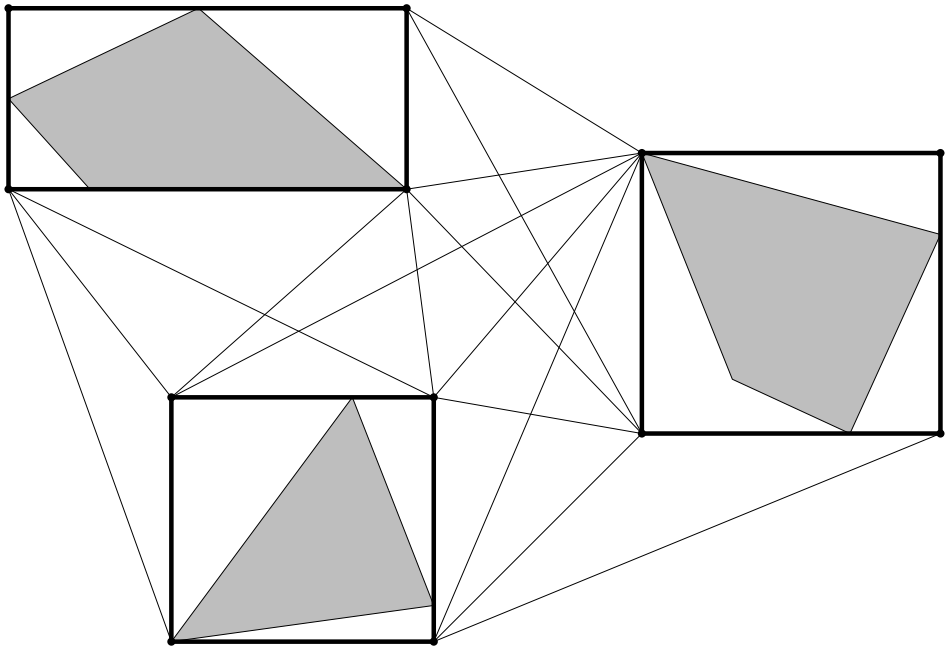

Figure 7.4: A bounding box visibility graph for three obstacles.

For the proof of Theorem 7.9, we compare the shortest path p_{st}^{vis} between a pair of nodes s and t in a visibility graph G_{Vis} with a path p_{st}^{BB} between s and t in the related bounding box visibility graph G_{BB}. Observe that p_{st}^{vis} goes along a sequence of polygons $(p_1, \ldots, p_k)$ from s to t. When p_{st}^{vis} goes from a polygon p_i to a polygon p_{i+1}, p_{st}^{vis} is x- and y-monotone for that part, because p_{st}^{vis} follows a direct line segment between p_i and p_{i+1}. The main idea is to construct a path p_{st}^{BB} in G_{BB} that has the same monotonicity properties as p_{st}^{vis} for every pair of consecutive visited polygons p_i and p_{i+1} of p_{st}^{vis}. We use a greedy routing strategy for G_{BB}. This strategy constructs paths with the same monotonicity properties as p_{st}^{vis}. The greedy strategy is called *Greedy Visibility Routing (GreViRo)* and we define it as follows:

Let p_{st}^{vis} be a shortest path between two points s and t in a visibility graph G_{Vis} that contains polygons with non-intersecting bounding boxes. The sequence of polygons visited by p_{st}^{vis} is written as $(p_1, \ldots, p_k)$ and the direct line segment walked by p_{st}^{vis} from polygon p_i to p_{i+1} is written as $p_{p_i p_{i+1}}^{vis}$. The intersection points of $p_{p_i p_{i+1}}^{vis}$ with $bb(p_i)$ and $bb(p_{i+1})$ are written as i_{p_i} and $i_{p_{i+1}}$ respectively. Let G_{BB} be the corresponding bounding box visibility graph. We consider the line segment $p_{p_i p_{i+1}}^{vis}$ of p_{st}^{vis}. GreViRo connects the two nodes v_{bb_i} and $v_{bb_{i+1}}$ of G_{BB} provided v_{bb_i} and i_{p_i} and also $i_{p_{i+1}}$ and $v_{bb_{i+1}}$ are visible from each other. Note that the path $(v_{bb_i}, i_{p_i}, i_{p_i+1}, v_{bb_{i+1}})$ has the same monotonicity properties as $p_{p_i p_{i+1}}^{vis}$.

GreViRo always chooses the node of a bounding box intersecting the face with nodes $v_{bb_i}, i_{p_i}, i_{p_i+1}$ and $v_{bb_{i+1}}$ that does not violate the monotonicity properties of $p_{p_i p_{i+1}}^{vis}$ and minimizes the distance to $p_{p_i p_{i+1}}^{vis}$ until it $v_{bb_{i+1}}$ is visible.

Figure 7.5 depicts a path construction of GreViRo. Observe that GreViRo is de-

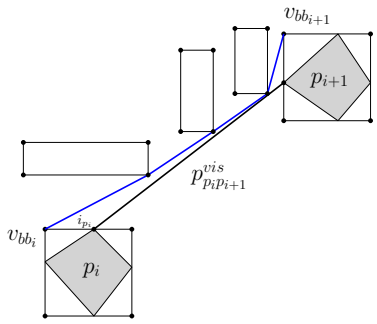

Figure 7.5: A path construction of GreViRo.

fined to fulfill the same monotonicity properties as p^{vis}_{st}. The following lemma states the correctness of GreViRo.

Lemma 7.11. Let v_{bb_i} and $v_{bb_{i+1}}$ be defined as described above. GreViRo constructs a path in G_{BB} between v_{bb_i} and $v_{bb_{i+1}}$.

We present the proof for this lemma later, see Lemma 7.19. Our routing strategy GreViRo lets us to construct a path in G_{BB} that has the same monotonicity properties as the original path in G_{Vis}. By applying GreViRo along every path segment connecting two different holes, we prove Theorem 7.16 respectively Theorem 7.9. Transferring the results of theorem 7.9 to the 2-*localized* Delaunay graph leads to the following corollary.

Corollary 7.12. Consider a $G = (V, E)$ which contains up to m holes. Between any pair of nodes s and t with $s, t \in V$ that do not lie in any bounding box, there exists a path p from s to t in G such that:

$$p \leq 5.66 \cdot d_{\text{UDG}}(s, t).$$

Figure 7.4 provides an example of a bounding box visibility graph. The high level proof idea is that we assume that the considered bounding box visibility graph contains a single bounding box. We assume that the starting location s and the target location t are not inside of the bounding box. The node-set V of the related bounding box visibility graph contains s, t, and the nodes of the bounding box: $V = \{s, t, bb_{t\ell}, bb_{tr}, bb_{b\ell}, bb_{br}\}$. We consider first a simple case, namely the case of one bounding box with proving Theorem 7.13 which states that there always exists a path of length at most $\sqrt{2} \cdot d_{\text{UDG}}(s, t)$ between s and t in the described setting.

Theorem 7.13. *Let $G = (V, E)$ be a bounding box visibility graph containing a single bounding box b with starting location $s \notin b$ and target location $t \notin b$. Then, there exists a path between s and t in G with length at most $\sqrt{2} \cdot d_{\text{UDG}}(s, t)$.*

For the proof of Theorem 7.13 we need a certain property of right triangles. We consider a right triangle with legs a and b and hypotenuse c and a ninety degree

angle between a and b. We prove that walking along a and b is not longer than a constant times walking along c. Lemma 7.14 deals with this property.

Lemma 7.14. Let a, b be the legs of a right triangle with hypotenuse c. Then, $a + b \leq \sqrt{2} \cdot c$.

Proof Pythagorean states $c^2 = a^2 + b^2 \iff c = \sqrt{a^2 + b^2}$. We compute the ratio x of $a + b$ and c and bound it.

$$\frac{a+b}{c} = \frac{a+b}{\sqrt{a^2+b^2}} \leq x \iff \frac{(a+b)^2}{a^2+b^2} \leq x^2$$

We have the equivalence when a, b, and c are greater than zero. With the binomial theorem, we obtain:

$$\frac{(a+b)^2}{a^2+b^2} = \frac{a^2+2ab+b^2}{a^2+b^2} = \frac{a^2+b^2}{a^2+b^2} + \frac{2ab}{a^2+b^2} = 1 + \frac{2ab}{a^2+b^2}$$

We analyze the properties of the latter addend and we have:

$$\begin{aligned} \frac{2ab}{a^2+b^2} \leq 1 &\iff 2ab \leq a^2 + b^2 \iff 0 \leq a^2 - 2ab + b^2 \\ &\iff 0 \leq (a-b)^2 \end{aligned}$$

Since quadratic numbers are always positive, our claim holds and we can finally plug all results together and finish the proof.

$$\frac{(a+b)^2}{a^2+b^2} = 1 + \frac{2ab}{a^2+b^2} \leq 1 + 1 = 2 = x^2 \iff \sqrt{2} = x$$

□

With the knowledge of Lemma 7.14 we prove Theorem 7.13.

Proof Assume $x(s) < x(t)$. We distinguish two different cases concerning the size of the bounding box. In Case 1, we have bounding boxes that fit completely into the bounding box around s and t. In Case 2, we have bounding boxes that exceed the bounding box around s and t. Consider the surrounding bounding box with nodes $s, t, u = (x(s), y(t))$, and $v = (x(t), y(s))$. **Case 1:** The bounding box of the obstacle is contained in the bounding box with nodes s, t, u, and v.
The worst case is that the bounding box of the hole and the bounding box with nodes s, t, u, and v coincide. Then, the shortest connection from s to t is the combination of the line segments $\overline{su}$ and $\overline{ut}$. The combination of $\overline{sv}$ and $\overline{vt}$ has about the same length. With Lemma 7.14, we obtain:

$$su + ut \leq \sqrt{2} \cdot st \leq \sqrt{2} \cdot d_{\mathrm{UDG}}(s, t).$$

We have the last inequality, because the Euclidean distance between s and t is the shortest possible length of a path between s and t. The distance of s and t in the Unit Disk Graph is larger or equal. If the two mentioned bounding boxes do not coincide, the shortest path among nodes of bounding boxes is smaller than a path along the comprising bounding box. Thus, we have $\sqrt{2} \cdot st$ as an upper bound for the length of the shortest path among nodes of a bounding box.

Case 2: The bounding box of the obstacle exceeds the bounding box with nodes s, t, u and v.

Then, we consider the cases in which $\overline{ut}$ and $\overline{sv}$ or $\overline{us}$ and $\overline{tv}$ are intersected by $\overline{st}$. Otherwise, we apply the same argumentation as in Case 1, because there is a path that is completely contained in the box with nodes s, t, u, and v. We assume that both $\overline{ut}$ and $\overline{sv}$ are intersected. We use a similar proof for the other case by turning the view about 90 degrees. Let $p_{y_{max}}$ be the highest point of the hole polygon and let $p_{y_{min}}$ be the lowest point, respectively. The shortest geometric connection between s and t passes either $p_{y_{max}}$ or $p_{y_{min}}$. We assume that the shortest geometric connection passes $p_{y_{max}}$. The shortest connection would be $\overline{sp_{y_{max}}}$ and $\overline{p_{y_{max}}t}$. We upper bound the length of this connection by giving the legs of two right triangles as maximal path length. Consider the points $s_{y_{max}} = (x(s), y(p_{y_{max}}))$ and $t_{y_{max}} = (x(t), y(p_{y_{max}}))$. The longest possible path over bounding box points would be $\overline{ss_{y_{max}}}$,$\overline{s_{y_{max}}t_{y_{max}}}$ and $\overline{t_{y_{max}}t}$. Since this path uses the legs of right triangles with hypotenuses $\overline{sp_{y_{max}}}$ and $\overline{p_{y_{max}}t}$, we know that the maximal length is at most $\sqrt{2} \cdot (sp_{y_{max}} + p_{y_{max}}t) \leq \sqrt{2} \cdot d_{\text{UDG}}(s, t)$. See Figure 7.6 for an example of both right triangles. The last inequality holds because the distance between s and t in the unit disk graph cannot be smaller than the shortest possible geometric connection between s and t. □

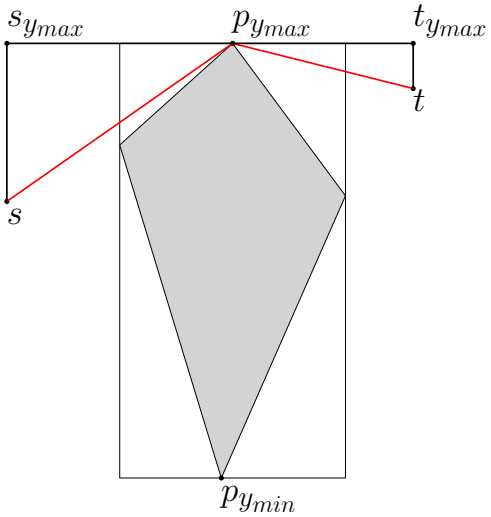

Figure 7.6: A visualization of case 2.

We have analyzed the maximal length of c-competitive paths we achieve for a single bounding box in bounding box visibility graphs. We combine these insights with our knowledge about virtual axes to bound the length of bounding box-paths in the 2-*localized* Delaunay graph. We use of Theorem 7.2 and obtain the following corollary.

Corollary 7.15. Consider a 2-localized Delaunay graph $G = (V, E)$ which contains a single hole with bounding box b. Between any pair of nodes s and t with $s, t \in V$ but $s, t \notin b$, there exists a path p from s to t that contains representatives of bounding boxes such that:

$$p \leq 5.66 \cdot d_{\text{UDG}}(s, t).$$

Additionally, there exists an online routing strategy that finds a path p_{on} from s to t such that:

$$p_{on} \leq 7.87 \cdot d_{\text{UDG}}(s, t).$$

Proof We reduce the 2-localized Delaunay graph with b to a bounding box visibility graph only containing s, t and all nodes of b. This contains a path between s and t with length at most $\sqrt{2} \cdot d_{\text{UDG}}(s, t)$. We use the nodes in G that are closest to the nodes of b as representatives for b. We apply virtual axis routing and thus obtain that between two adjacent representatives of b a path of length at most 3.996 times their Euclidean distance exists. In 2-*localized* Delaunay graphs which contain a single hole with bounding box b there is a path between any pair of nodes $s, t \notin b$ of length at most $\sqrt{2} \cdot 3.996 \cdot d_{\text{UDG}}(s, t) \leq 5.66 \cdot d_{\text{UDG}}(s, t)$. This proves the corollary. For the online routing strategy, we find paths of length $\sqrt{2} \cdot 5.66 \cdot d_{\text{UDG}}(s, t) \leq 7.87 \cdot d_{\text{UDG}}(s, t)$.□

Let us continue with considering multiple non-intersecting bounding boxes, see also Theorem 7.9.

Theorem 7.16. *Let $G_{BB} = (V_{BB}, E)$ be a bounding box visibility graph that contains multiple non-intersecting bounding boxes and a source- and a target-location s and t. There exists a path p_{st}^{BB} between s and t in G_{BB} with:*

$$p_{st}^{BB} \leq \sqrt{2} \cdot d_{\text{UDG}}(s, t).$$

We are left with proving the correctness of GreViRo, see also lemma 7.11.

Lemma 7.17. Let v_{bb_i} and $v_{bb_{i+1}}$ be defined as described above. GreViRo constructs a path in G_{BB} between v_{bb_i} and $v_{bb_{i+1}}$.

Proof We assume that $p_{p_i p_i+1}^{vis}$ is increasing x- and increasing y-monotone. A similar proof can be used for all other cases by turning the view $90, 180$ and 270 degrees. Turning the view 90 degrees yields an increasing x- and decreasing y-monotone

path, 180 degrees yields a decreasing x- and decreasing y-monotone path, and 270 degrees yields a decreasing x- and increasing y-monotone path.
We prove by contradiction that GreViRo constructs a path between v_{bb_i} and $v_{bb_{i+1}}$. Assume GreViRo has reached a node v_{bb_j}. It cannot go further because $v_{bb_{i+1}}$ is not visible. All other visible nodes violate increasing x- and y-monotonicity. Since $v_{bb_{i+1}}$ is not visible from v_{bb_j}, there is a bounding box bb_{j+1} that is intersected by the line segment $\overline{v_{bb_j} v_{bb_{i+1}}}$. We consider a visible node $v_{bb_{j+1}}$ of bb_{j+1}. According to our assumption, $x(v_{bb_j}) < x(v_{bb_{j+1}})$ and $y(v_{bb_j}) > y(v_{bb_{j+1}})$. Now consider the node $v_{bb_{j-1}}$ that has been visited before proceeding to v_{bb_j}. According our assumption, GreViRo gets stuck at node v_{bb_j}. Hence, $x(bb_{j-1}) < x(bb_j)$ and $y(bb_{j-1}) < y(bb_j)$. We observe that $v_{bb_{j+1}}$ has already been visible from $v_{bb_{j-1}}$. Further observe that there cannot be a third bounding box intersecting the line segment $\overline{v_{bb_{j-1}} v_{bb_{j+1}}}$ because nodes of this bounding box would have been preferred by GreViRo to $v_{bb_{j+1}}$ because these are closer to $p^{vis}_{p_i p_{i+1}}$. Hence, GreViRo would have chosen the node $v_{bb_{j+1}}$ instead of v_{bb_j} which is a contradiction to our assumption. We refer to Figure 7.7 for an example of the contradiction. □

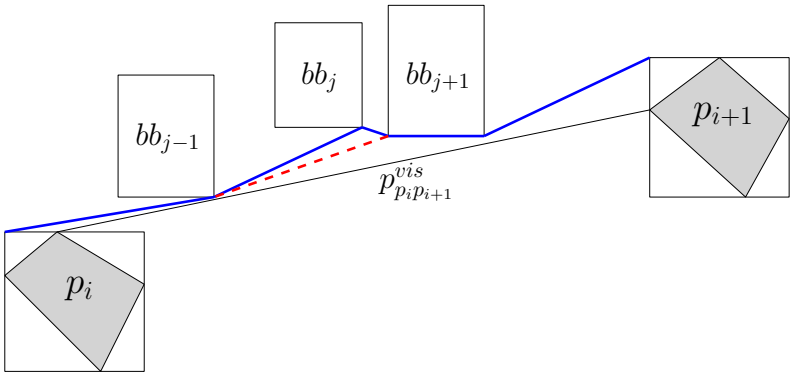

Figure 7.7: The contradiction described in the proof of Lemma 7.17. The contradicting path is marked by a thick blue line. The reader can see that bb_{j+1} has already been visible from bb_{j-1}.

GreViRo is a tool that lets us construct a path in G_{BB} that fulfills the same monotonicity properties as the original path in G_{Vis}. To prove Theorem 7.16, we split it into two lemmas. We assume that p^{vis}_{st} is x- and y-monotone (Lemma 7.18). Later on, we drop this assumption in Lemma 7.19 and assume that p^{vis}_{st} is either x- or y-monotone but not both. Finally, we drop any of the monotonicity assumptions and prove Theorem 7.16 with the Lemmas 7.18 and 7.19.

Lemma 7.18. If p^{vis}_{st} is x- and y-monotone, then there exists a path p^{BB}_{st} between s and t in G_{BB} with length at most $\sqrt{2} \cdot d_{\text{UDG}}(s,t)$.

Proof We construct a path p^{BB}_{st} in G_{BB} that is also x- and y-monotone such that we conclude that p^{BB}_{st} does not go a longer distance either in horizontal or in vertical direction than p^{vis}_{st}. We assume p^{vis}_{st} is both increasing x- and y-monotone. The proofs for the other cases can be obtained by turning the view by 90, 180 and 270

degrees. Let us consider the sequence of hole polygons $(p_1, \ldots, p_k)$ which is visited by the shortest path p_{st}^{vis}. We observe that when walking from p_i to p_{i+1}, p_{st}^{vis} intersects either the lower edge of $bb(p_{i+1})$ or the left edge of $bb(p_{i+1})$. Note that the edges of $bb(p_{i+1})$ are not part of G_{Vis}. Instead they are used to understand the path construction in G_{BB}. According to the monotonicity assumption, p_{st}^{vis} intersects either the upper edge of $bb(p_{i+1})$ or the right edge of $bb(p_{i+1})$ when proceeding to p_{i+2}.
The detailed path construction works as follows: Assume the path has $bb(p_i)$ and p_{st}^{vis} is heading to polygon p_{i+1} next. There are two different cases to consider. Either the left or the lower edge of $bb(p_{i+1})$ is intersected by p_{st}^{vis}. When the left edge is intersected, according to the monotonicity assumption, we know that the upper edge has to be intersected afterward. Hence, we use the top left node of p_{i+1} as the next point of our path-construction. The same argumentation holds if p_{vis} intersects the lower edge of $bb(p_{i+1})$. According to the monotonicity assumption, we know that the right edge of $bb(p_{i+1})$ has to be intersected when p_{st}^{vis} proceeds to p_{i+2}. Hence, we walk to the lower right node of $bb(p_{i+1})$. The next node of our path construction is:

- Case 1: $bb_{t\ell}(p_{i+1})$ if p_{st}^{vis} intersects $\overline{bb_{t\ell}(p_{i+1})bb_{b\ell}(p_{i+1})}$.
- Case 2: $bb_{br}(p_{i+1})$ if p_{st}^{vis} intersects $\overline{bb_{b\ell}(p_{i+1})bb_{br}(p_{i+1})}$.

The path visits the top left or the bottom right node of a bounding box. The construction described above yields 4 different paths from p_i to p_{i+1}:

1. Path from $bb_{t\ell}(p_i)$ to $bb_{t\ell}(p_{i+1})$
2. Path from $bb_{t\ell}(p_i)$ to $bb_{br}(p_{i+1})$
3. Path from $bb_{br}(p_i)$ to $bb_{t\ell}(p_{i+1})$
4. Path from $bb_{br}(p_i)$ to $bb_{br}(p_{i+1})$

There is no guarantee that the bounding box nodes of p_i and those from p_{i+1} are visible from each other.
Each case is a valid input for GreViRo. Hence, we apply GreViRo for every case that coonects $bb(p_i)$ and $bb(p_{i+1})$ according to the path construction. With Lemma 7.17, we have that p_{st}^{BB} is also increasing x- and y-monotone. We apply the construction and GreViRo for every sub-path from p_i to p_{i+1} for all $i \in 1, \ldots, k-1$ and we have an increasing x- and y-monotone path that passes all polygons that are visited by p_{st}^{vis}. The path between s and p_1 and also the path between p_k and t are constructed in the same way. Hence, p_{st}^{BB} goes neither a longer vertical nor horizontal distance than p_{st}^{vis}.
We compare p_{st}^{BB} to p_{st}^{vis} and use a right triangle with p_{st}^{vis} as length of the hypotenuse c, see Figure 7.9 as an example. The legs of the right triangle are the parts of p_{st}^{vis} that are walked in horizontal and in vertical direction. Hence, we know that the length of p_{st}^{BB} is at most the sum of both legs. Thus, we bound the length of p_{st}^{BB} by $\sqrt{2} \cdot p_{st}^{vis} \leq \sqrt{2} \cdot d_{\text{UDG}}(s,t)$. $\square$

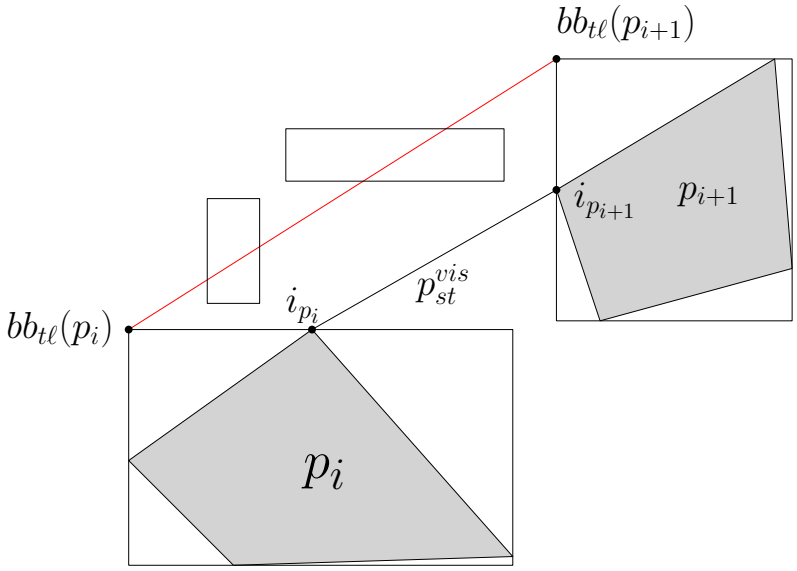

Figure 7.8: Comparison of p_{st}^{vis} and a potential bounding box path. p_{st}^{vis} uses a direct visibility line whereas p_i and p_{i+1} are not visible from each other.

Lemma 7.19. If the shortest path p_{st}^{vis} between s and t in G_{Vis} is either x- or y-monotone, then there exists a path p_{st}^{BB} between s and t in G_{BB} with length at most $\sqrt{2} \cdot d_{\text{UDG}}(s,t)$.

Proof The main proof idea is that between any visited pair of polygons p_i and p_{i+1} the path p_{st}^{vis} is x- and y-monotone. The path construction obtains a path that fulfills the same monotonicity property as p_{st}^{vis} for any pair of succeeding visited hole polygons. We assume p_{st}^{vis} is increasing x-monotone but it is not y-monotone. A 90 degrees turn gives the proof for the other case. For the proof of Lemma 7.19 we extend the path construction from the proof of Lemma 7.18 as follows: The path arrived at the bounding box of hole polygon p_i and we want to reach to polygon p_{i+1} as a next step. There are also other cases which have to be considered for this setting. (Case 3) says that p_{st}^{vis} intersects the left edge of $bb(p_{i+1})$ and afterward the lower edge of $bb(p_{i+1})$ because we do not assume y-monotonicity. In this case, choose p_{i+1} as the next node. It might happen that p_{st}^{vis} intersects the upper edge of $bb(p_{i+1})$ and the right edge later (Case 4). In this case, p_{i+1} is chosen.
There is a further case that we have to consider (Case 5). Because we do not assume y-monotonicity anymore, it might happen that p_{st}^{vis} intersects either the upper or the lower edge of $bb(p_{i+1})$ and leaves $bb(p_{i+1})$ afterward through the same edge without intersecting any other edge of $bb(p_{i+1})$. We do not choose a node of $bb(p_{i+1})$ because there is no guarantee that that there is an x- and y-monotone path from p_{i+1} to p_{i+2} thereafter, see Figure 7.10 as an example.
So we avoid any node of $bb(p_{i+1})$ and walk towards a node until the first intersection point of p_{st}^{vis} with $bb(p_{i+1})$, which is denoted as $i_{p_{i+1},1}$, is visible. This node will be the start point for the sub-path between p_{i+1} and p_{i+2}. We see the new cases like this: Assume we have already covered the sub-path to p_i and p_{st}^{vis} is going to p_{i+1}

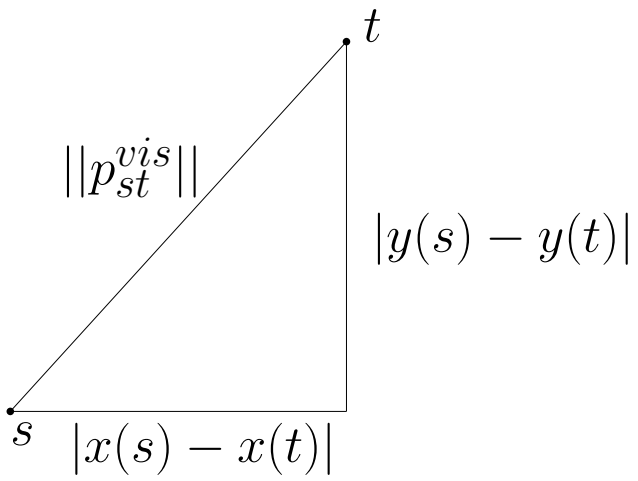

Figure 7.9: A triangle comparing p_{st}^{vis} and p_{st}^{BB}. The length of p_{st}^{BB} is at most the sum of the legs of the depicted right triangle.

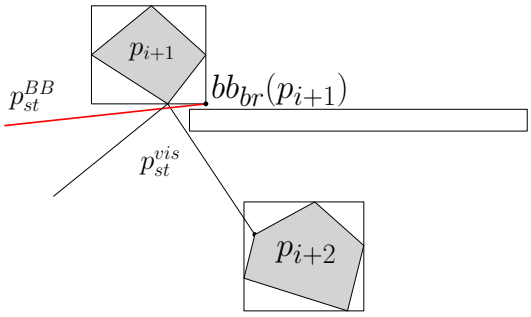

Figure 7.10: An invalid path construction for Case 5. If p_{st}^{BB} chooses p_{i+1}, p_{st}^{BB} has to cover a longer distance in horizontal direction than p_{st}^{vis} when walking from p_{i+1} to p_{i+2}.

next. The next node of the path is:

- Case 3: $bb_{tr}(p_{i+1})$ if p_{st}^{vis} intersects $\overline{bb_{t\ell}(p_{i+1})bb_{tr}(p_{i+1})}$ and afterward $\overline{bb_{tr}(p_{i+1})bb_{br}(p_{i+1})}$
- Case 4: $bb_{b\ell}(p_{i+1})$ if p_{st}^{vis} intersects $\overline{bb_{t\ell}(p_{i+1})bb_{b\ell}(p_{i+1})}$ and afterward $\overline{bb_{b\ell}(p_{i+1})bb_{br}(p_{i+1})}$
- Case 5: a node of $bb(p_j)$ such that $i_{p_{i+1},1}$ is visible, if p_{st}^{vis} intersects $\overline{bb_{t\ell}(p_{i+1})bb_{tr}(p_{i+1})}$ or $\overline{bb_{b\ell}(p_{i+1})bb_{br}(p_{i+1})}$ twice without intersecting any other edge of $bb(p_{i+1})$.

Figures 7.11, 7.12 and 7.13 present examples of each individual case.
Case 3 has similar arguments to those we have used for the path construction in the proof of Lemma 7.18. Case 4 needs a few more arguments, because p_{st}^{vis} does not fulfill y-monotonicity. This implies that p_{st}^{vis} is increasing y-monotone from p_i to p_{i+1}. We construct a path with GreViRo for two sub-paths between p_i and p_{i+1}. The first sub-path is increasing x- and increasing y-monotone. The second path stays increasing x-monotone and changes to decreasing y-monotonicity.

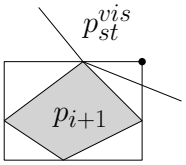

Figure 7.11: Visualization of Case 3.

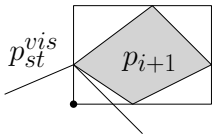

Figure 7.12: Visualization of Case 4.

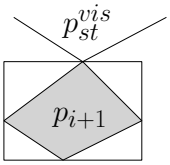

Figure 7.13: Visualization of Case 5.

Apply GreViRo from p_{st}^{vis} until it reaches p_{i+1}, or another node v is chosen with $y(v) > y(i_{p_{i+1}})$. In the second case, GreViRo is applied to the line segment $\overline{i_{p_{i+1}} p_{i+1}}$. This might happen at a node which is visible from $i_{p_{i+1}}$. Thus, this is a valid input for GreViRo for the second sub-path. By Lemma 7.17, the first part of the path is increasing x- and increasing y-monotone and the second path is increasing x- and decreasing y-monotone. Thus, p_{st}^{BB} fulfills the same monotonicity properties as p_{st}^{vis} between p_i and p_{i+1}. The proof for the other start nodes is similar to the above proof and thus it is omitted here.
No, we prove the path existence for case 5. Let us assume that from p_i to p_{i+1} case 5 occurs for the first time and hence we are either at $bb_{tl}(p_i), bb_{tr}(p_i), bb_{bl}(p_i)$, or $bb_{br}(p_i)$. We assume that we are positioned at p_i. We also assume that p_{st}^{vis} intersects the lower edge of bb_{i+1}. By GreViRo, we eventually reach a node of $bb(p_j)$ such that $i_{p_{i+1},1}$ is visible. Hence, we achieve an increasing x- and increasing y-monotone sub-path to a node of $bb(p_j)$. We consider now the part between $bb(p_j)$ until we can walk along p_{st}^{vis} again, see Figure 7.14 as an example.

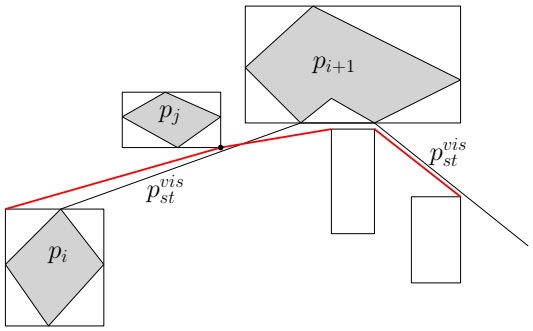

Figure 7.14: GreViRo example for Case 5. After reaching the node of p_j, GreViRo has to be applied until $i_{p_{i+1},2}$ is visible.

Our aim is to introduce another sub-path. The construction works as follows. Start at $bb(p_j)$ and try to reach a node of $bb(p_{j'})$ such that $i_{p_{i+1},2}$ is visible. After reaching that point, we go along p_{st}^{vis} with the greedy strategy. That this is a valid input for GreViRo. By applying GreViRo, we have an increasing x- and increasing y-monotone sub-path between $bb(p_j)$ and $bb(p_{j'})$. Finally, we plug all these sub-paths together. The other cases are similar and are therefore omitted here. Thus, we have: Each sub-path of p_{st}^{BB} does not cover a longer distance in horizontal or vertical direction than p_{st}^{vis}. We use the right triangle of Lemma 7.18 again, see

Figure 7.9). Consequently:

$$p_{st}^{BB} \leq \sqrt{2} \cdot p_{st}^{vis} \leq \sqrt{2} \cdot d_{\text{UDG}}(s,t).$$

□

We prove the correctness of Theorem 7.16.

Proof We drop the last monotonicity assumption and obtain a few more cases. We assume that the path brought us to $bb(p_i)$ and we want to go to $bb(p_{i+1})$ as the next step. The new case (Case 6) is that the right edge of bounding box $bb(p_{i+1})$ is intersected and afterward the lower edge. In this case we go to p_{i+1}. The other new case (Case 7) is that the right edge of $bb(p_{i+1})$ is intersected first and afterward the upper edge. We go to p_{i+1}. Case 8 is similar to Case 5 of Lemma 7.19. In this case, either the left or the right edge of $bb(p_{i+1})$ is intersected twice without intersecting any other edge of $bb(p_{i+1})$. We use the same idea for the proof of Lemma 7.19. We go as close as possible to $bb(p_{i+1})$ such that $i_{p_{i+1},1}$ is visible. Thus, the path construction chooses the following node of $bb(p_{i+1})$:

- Case 6: $bb_{br}p_{(i+1)}$ if p_{st}^{vis} intersects $\overline{bb_{tr}p_{(i+1)}bb_{br}p_{(i+1)}}$ first and afterward $\overline{bb_{bl}p_{(i+1)}bb_{br}p_{(i+1)}}$.
- Case 7: $bb_{tr}p_{(i+1)}$ if p_{st}^{vis} intersects $\overline{bb_{tr}p_{(i+1)}bb_{br}p_{(i+1)}}$ first and afterward $\overline{bb_{tr}p_{(i+1)}bb_{br}p_{(i+1)}}$.
- Case 8: A node of $bb(p_j)$ such that $i_{p_{i+1},1}$ is visible, if p_{st}^{vis} intersects $\overline{bb_{tr}(p_{i+1})bb_{br}(p_{i+1})}$ or $\overline{bb_{t\ell}(p_{i+1})bb_{b\ell}(p_{i+1})}$ twice without intersecting any other edge of $bb(p_{i+1})$.

Figures 7.15, 7.16 and 7.17 provide examples for each case. All of the cases above

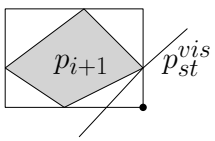

Figure 7.15: Visualization of case 6.

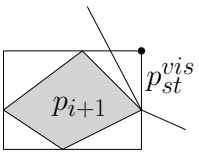

Figure 7.16: Visualization of case 7.

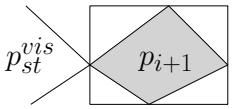

Figure 7.17: Visualization of case 8.

are either point or axis reflections of the path constructions we have proven for Lemmas 7.18 and 7.19. We construct a path that fulfills the same monotonicity properties as p_{st}^{vis} between each pair of succeeding hole polygons p_i and p_{i+1} that are visited by p_{st}^{vis}. Hence, there is a path p_{st}^{BB} in G_{BB} with:

$$p_{st}^{BB} \leq \sqrt{2} \cdot d_{\text{UDG}}(s,t). \; \square$$

Consider m non-intersecting bounding boxes in a 2-localized Delaunay graph and assume that each node of a bounding box is represented by the closest node of the 2-localized Delaunay graph. Thus, we apply virtual axis routing between any two adjacent nodes on the path p_{st}^{BB}. We obtain a path in that length at most $\sqrt{2} \cdot 3.996$ $d_{\mathrm{UDG}}(s,t) = 5.66 \cdot d_{\mathrm{UDG}}(s,t)$. For online routing, we have a path of length at most $\sqrt{2} \cdot 5.66$ $d_{\mathrm{UDG}}(s,t) = 7.87 \cdot d_{\mathrm{UDG}}(s,t)$.

7.3 Routing Algorithm

For the routing algorithm *Bounding Box Routing* (*BBR*), assume the source s and the destination t are outside of bounding boxes. The source node s sends the message via the MixedChordArc algorithm into the direction of the target node t. If the message arrives at a hole node, the message is directed to the closest representative of the corresponding bounding box. The representative computes a path from its position to t in its modified bounding box visibility graph. This is is the path to route the message to t. The entire path information has not to be contained in the message but only the next destination. The node at the destination of the message computes the next destination of the message and forward it. Along every edge of the bounding box visibility graph, MixedChordArc is used to forward the message. *BBR* fullfils the following routing properties:

Theorem 7.20. *BBR finds paths between a source s and a target t outside of bounding boxes with length at most* $18.55 \cdot d_{UDG}(s,t)$.

Proof Based on the results of Section 7.2, online routing via the path in the bounding box visibility graph gives a path of length $7.87 \cdot d_{UDG}(s,t)$. BBR uses initially MixedChordArc. This might result in a detour. The detour has length at most $3 \cdot 3.66 \cdot d_{UDG(s,t)} = 10.68 \cdot d_{UDG}(s,t)$, because the path until reaching a hole has length at most $3.66 \cdot d_{UDG}(s,t)$. The factor 3 means the detour the routing could take when it reaches it hole. We have the same for the path to the closed representative of a bounding box. This is a detour of $10.68 \cdot d_{UDG}(s,t)$. The complete path is $10.68 + 7.87 = 18.55$-competitive. □

8 Intersection of Hole Abstractions

In the previous chapters, we considered convex hulls and bounding boxes as abstractions for holes. We built a cellular infrastructure on top and proved it to give information about competitive routing paths in the ad hoc network. However, we considered non-intersecting convex hulls and bounding boxes.
In this chapter, we consider also cases in which hole abstractions intersect, i.e., the intersection of bounding boxes. First, we point out the problem of routing through intersections. To make competitive routing through bounding boxes possible, when source s and target t are outside of bounding boxes, we present a suitable cellular infrastructure, the modified bounding box visibility graph in Section 8.1. We present PairwiseIntersectionChord (PIC) in Section 8.2 for pairewise intersecting bounding boxes, followed by Section 8.3 for competitive routing through multiple intersecting bounding boxes.
The results of this chapter are published in [5] and [6].

8.1 Competitive Paths via Intersecting Bounding Boxes

Intersecting bounding boxes of holes lead to new challenges, i.e., what if the shortest path between nodes s and t leads through an area in which two or more bounding boxes intersect? One problem is that nodes of bounding boxes can be in holes, which is given by an example in the following figure.

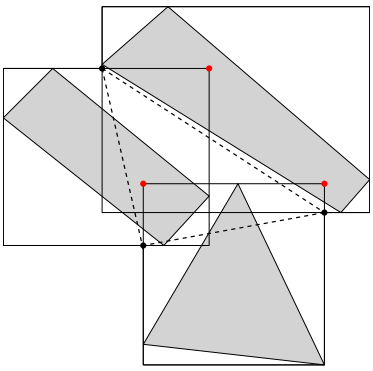

Figure 8.1: Multiple intersecting bounding boxes with nodes lying in holes. The outer intersection points of the bounding boxes are connected in a clique.

Maybe all the nodes of bounding boxes are inside of intersecting bounding boxes and lie in holes and thus we cannot receive any information out of these nodes. To

avoid this situation, we drop all of these nodes and only keep the nodes located on the outer boundary of all intersecting bounding boxes.

Routing is easy as long as the shortest path avoids such situations. We apply Theorem 7.16 and obtain a $\sqrt{2}$-competitive path.

The shortest path that goes through an area of two or more intersecting bounding boxes can be very complex. We cannot find c-competitive paths by only using information from bounding boxes. Thus, we better make use of the information contained in the bounding box visibility graph.
Use the outer boundary of where multiple bounding boxes intersect. On this outer boundary, there are nodes of bounding boxes and intersection points of bounding boxes. We call these points *outer intersection points*. When a shortest path leads through the area where at least two bounding boxes intersect, an outer intersection point has to be passed in vertical and also in horizontal direction.

We modify bounding box visibility graphs as follows: When two or more bounding boxes intersect, we keep the nodes which lie on the outer boundary of and we drop all nodes which are inside. We insert all outer intersection points into the node set. Outer intersection points are connected in a clique. The weight of an edge that connects outer intersection points o_1 and o_2 is deonted as the length of the shortest path that is in the corresponding visibility graph connecting o_1 and o_2 inside the intersection area. We call this construction *modified Bounding Box Visibility Graph*. Figure 8.1 gives a good example of the clique of outer intersection points.

8.2 Pairwise Intersecting Bounding Boxes

In this section, we provide an approach for routing through hole abstractions. Assume the intersection of two bounding boxes as hole abstractions. We assume that the convex hulls of their radio holes do not intersect. We present PairwiseIntersectionChord (PIC) that routes $\sqrt{2}$-competitive through the bounding box intersection between two outer intersection points of pairwise intersecting bounding boxes. With the competitive constant of PIC, we choose the weights of edges of the modified bounding box visibility graph as follows: Let e be an edge between two outer intersection points o_1 and o_2. We add $\sqrt{2} \cdot \|o_1\, o_2\|$ as weight to edge e.

We obtain the following constant for the existence of a routing path through the intersection of two bounding boxes between s and t.

Theorem 8.1. *If the modified bounding box visibility graph contains pairwise intersecting bounding boxes and the edge weight between outer intersection points is at most $\sqrt{2}$ times the length of the shortest path between these points, BBR finds*

paths between a source s and a target t outside of bounding boxes with length at most $28.83 \cdot d_{UDG}(s,t)$.

Proof: We use Theorem 7.20. We also have a detour of $10.68 \cdot d_{UDG}(s,t)$. We apply Corollary 8.2. The path is $\sqrt{2} \cdot 12.83 \leq 18.15$-competitive, because of the additional $\sqrt{2}$ obtained by PIC. Hence, the entire path has length at most $10.68 + 18.15 = 28.83 \cdot d_{UDG}(s,t)$. □

PIC ALGORITHM: PIC routes along the convex hulls of holes that belong to the outer intersection points. Convex hulls give the routing direction through the pairwise intersection of bounding boxes. The PIC algorithm has the following 3 steps. We call the source outer intersection point by bb_{int1} and the destination point bb_{int4}.

Assume that the bounding boxes looks like in Figure 8.2 and denote the upper left

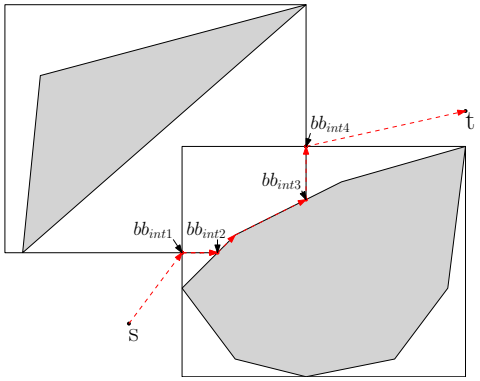

Figure 8.2: An exemplary path constructed by PIC.

bounding box which belongs to a polygon p_1 as the *upper* bounding box and the other bounding box which belongs to a polygon p_2 as the lower bounding box. We route from the left outer intersection point to the right one. All other cases are rotated cases and are handled in the same way.

1. bb_{int1} sends a message m along the edge $\overline{bb_{b\ell}(p_1)bb_{br}(p_1)}$ of the upper bounding box until m reaches the intersection point bb_{int2} of $\overline{bb_{b\ell}(p_1)bb_{br}(p_1)}$ and a convex hull.

2. Then from bb_{int2}, m is sent along the convex hull edges until it m arrives at the point bb_{int3}, which is the intersection point between a convex hull edge and $\overline{bb_{tr}(p_2), bb_{br}(p_2)}$.

3. bb_{int3} sends m to the destination point bb_{int4}, which denotes the intersection point of $\overline{bb_{tr}(p_2), bb_{br}(p_2)}$ from the upper bounding box and $\overline{bb_{t\ell}(p)bb_{tr}(p)}$ from the lower bounding box.

To route m along the edges as in the three steps above, we use MixedChordArc.

Theorem 8.2. *PIC finds paths path between two outer intersection points o_1 and o_2 of pairwise intersecting bounding boxes with length at most $\sqrt{2} \cdot \|o_1\, o_2\|$.*

Proof: To route between two outer intersection points o_1 and o_2 of pairwise intersecting bounding boxes, the routing path is obtained in one direction. This direction goes from o_1 to o_2. The paths never increases, neither in x- nor in y-direction. Thus, PIC provides paths between two outer intersection points o_1 and o_2 of pairwise intersecting bounding boxes with length at most $\sqrt{2} \cdot \|o_1\, o_2\|$. □

8.3 Multiple Intersecting Bounding Boxes

The algorithm for multiple intersecting bounding boxes is similar to the algorithm for the case with non-intersecting bounding boxes. We consider modified bounding box visibility graphs for the computation of short paths. If we know a weight between outer intersection points of the modified bounding box visibility graph which has a length at most c times the length of an optimal path between the outer intersection points, then we can show the following theorem:

Theorem 8.3. *If the modified bounding box visibility graph contains an edge weight between outer intersection points which is at most c times the length of the shortest path between these points, BBR finds paths between a source s and a target t outside of bounding boxes with length at most* $(10.68 + c \cdot 12.83) \cdot d_{UDG}(s,t)$.

Proof: The proof is similar to the proof of Theorem 8.1. □

Theorem 8.1 is a special case of Theorem 8.3 for $c = \sqrt{2}$. We do not know such an edge weight for outer intersection points.

Because the computation of c-competitive path between outer intersection points can be very complex, we use the following strategy here: We use GOAFR+ between outer intersection points. Let o_1 and o_2 be two outer intersection points of the same hole. Let $\alpha \cdot \|o_1 o_2\|^2$ be the weight for the edge $\{o_1, o_2\}$ in the modified bounding box visibility graph. We do this, because GOAFR+ is in worst cases quadratic competitive to a shortest possible path. Good values for α provide the simulations.

8.4 Simulation Results

We show that this approach is better instead of using GOAFR+ for the entire path. We show suitable values for α. All in all, we use the same procedure as for non-intersecting bounding boxes. The difference is the weight of edges between outer intersection points in the modified bounding box visibility graph. Whenever an edge between outer intersection points is part of a path, GOAFR+ is used instead of MixedChordArc.

We compare our routing strategy with existing strategies and prove that ours outperform these strategies both for intersecting and non-intersecting bounding boxes. By Outperforming, we mean that the path length found by *BBR* is significantly shorter than those found by strategies that do not consider any global knowledge about holes. To compare our results to earlier results properly, we choose the same experiment setup as in [31].

The simulations are done on randomly generated unit disk graph. Nodes are positioned randomly and uniformly on a square with 20 units side length. The number

of nodes depends on the network density. Density means the number of nodes per unit disk. Because we are only interested in connected networks, the lowest density value we use is 4.5. Even for the value 4.5 not all of the randomly and uniformly generated networks are connected. We run the simulations only on connected networks. For density values between 4.5 and 20, we generate 2000 networks and place a source node s and a target node t uniformly at random. The simulations are done on a custom simulation environment. We compute the following performance value for every chosen algorithm:

$$perf_A(N, s, t) = \frac{\|p_A(N, s, t)\|}{\|p_{opt}(N, s, t)\|}$$

$|p_A(N, s, t)\|$ represents the length of the path found by algorithm A and $\|p_{opt}(N, s, t)\|$ stands for the length of an optimal path contained in the network N. Before switching the results of the simulations, we introduce compared algorithms.

8.5 Algorithms for Bounding Box Intersections

In the following, we introduce the algorithms.

OAFR Let $\mathscr{E}(c)$ be the ellipse with foci s and t and the size of its major axis is c. Initially, set $\mathscr{E}$ to $2 \cdot \|st\|$. Explore the boundary of the face F that is intersected by $\overline{st}$ in one direction. Upon reaching the boundary of $\mathscr{E}$, switch the direction. Once $\mathscr{E}$ is hit a second time, proceed to the node closest to t. If t is not yet reached, double the length of $\mathscr{E}'s$ major axis and start again.

GOAFR Initially, set $\mathscr{E}$ to $2 \cdot \|st\|$. Apply greedy routing until either arriving at t or a local minimum m. Execute OAFR on the first face only. The size of $\mathscr{E}$ is doubled as long as necessary. Terminate if OAFR reaches t. Otherwise, approach to the node closest to t found by OAFR and start again.

GOAFR$_{\mathsf{FC}}$ This works like GOAFR but falls back to greedy routing as soon as a closer node to t is found in the OAFR-phase [31].

GOAFR+ GOAFR+ works similar to GOAFR but the size of the major axis of $\mathscr{E}$ is not doubled. Let m be the node where GOAFR switches from greedy routing to OAFR. Now, GOAFR continues and counts in a variable p the number of nodes closer to t than u_i and q the number of other nodes. Let $r_{\mathscr{E}}$ be the major axis of $\mathscr{E}$. The new $r_{\mathscr{E}}$ is $p \cdot r_{\mathscr{E}}$.

GPSR GPSR stands for Greedy Perimeter Stateless Routing. It works exactly as GOAFR_{FC} without a bounding ellipse.

Non-intersecting Bounding Boxes The following figure shows the simulation results for the case of non-intersecting bounding boxes.

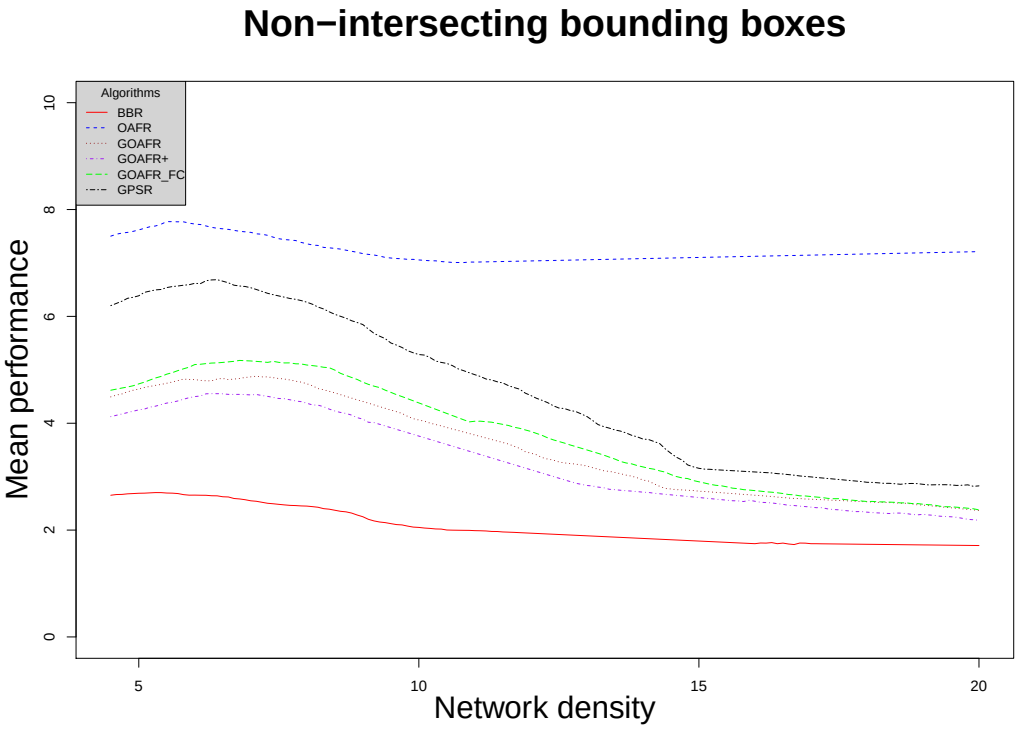

Figure 8.3: Visualization of the performance of the algorithms in case of non-intersecting bounding boxes. The y-axis represents the performance value of the algorithms. The x-axis represents the network density. The lowest red line represents the performance of BBR.

For each density value, the mean performance value of each algorithm, is defined as

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

is plotted. In our simulations, $k = 2000$. We see that for each density value, BBR performs better than any of the above mentioned algorithms. Even in cases where many holes exists, the density values are about 4.5. Avoiding holes around their bounding boxes performs a way better than any strategy that only considers local knowledge. Also, for scenarios with few holes and high density values, BBR still performs better than, for example, GOAFR+. All the mentioned algorithms have a small peak at density values around 8.

Intersecting Bounding Boxes To get a better impression about the performance for the case of intersecting bounding boxes, we choose the source destination pairs

whose shortest path goes through an area of intersecting bounding boxes. The following figure shows a plot of the mean performance of all algorithms, depending on the density value of the network.

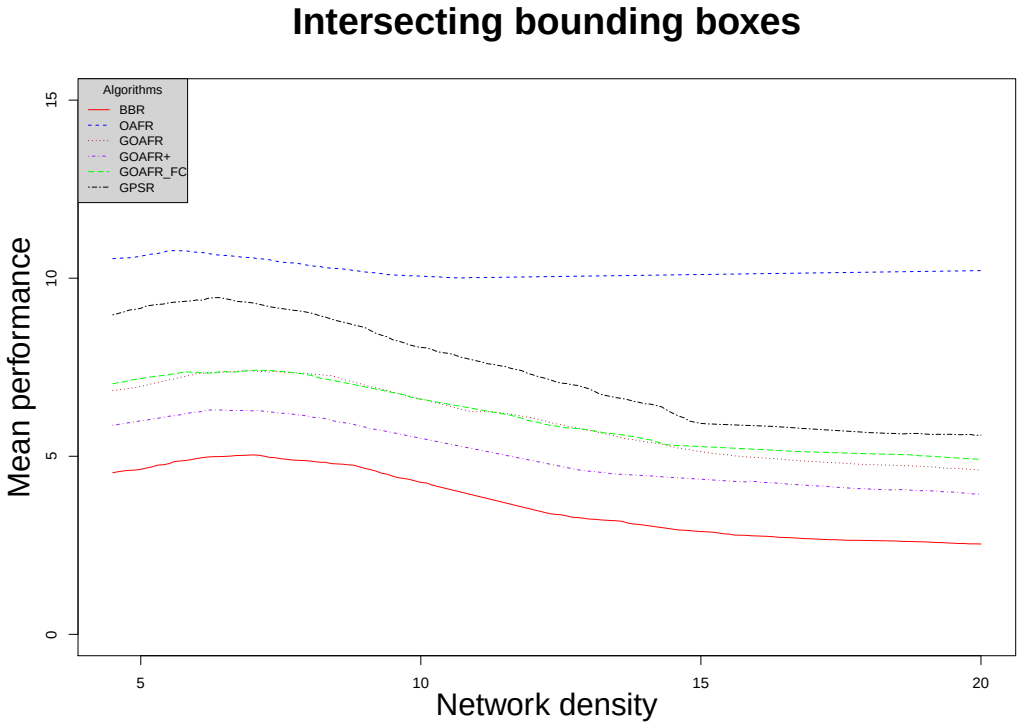

Figure 8.4: Visualization of the performance of the algorithms in case of intersecting bounding boxes. The y-axis represents the performance value of the algorithms. The x-axis represents the network density. The lowest red line represents the performance of BBR.

Here, we can see that BBR performs significantly better than all of the other described approaches. However, the line of BBR is closer to the line of GOAFR+ compared to non-intersecting bounding boxes. This might be because BBR uses GOAFR+ between outer intersection points.

9 Gaining Routing Information

In this chapter, we present the distributed detection of hole nodes. Further, we present the computation of convex hulls out of a hole, as well as the calculation for bounding boxes. Finally, we present the dissemination of convex hull and bounding box locations. We use the following approach:

Given is the unit disk graph. To determine the radio holes of the wireless ad hoc network, we compute the 2-localized Delaunay graph. This computation needs $\mathcal{O}(1)$ communication rounds in the $\mathcal{LOCAL}$ model. The 2-localized Delaunay graph allows the nodes to detect whether they are at the boundary of a radio hole. Nodes at the boundary form a ring.

We then design a distributed algorithm that computes the convex hull of the hole ring of n nodes in expected $\mathcal{O}(\log n)$ communication rounds, see the following figure.

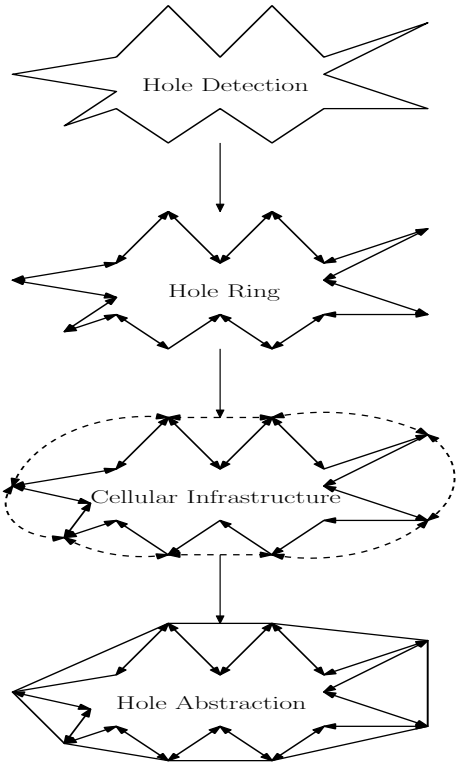

Figure 9.1: Information dissemination.

We introduce the nodes of the convex hulls to each other so that they form a clique. This helps them to compute c-competitive paths for all source-destination pairs that are outside of a convex hull. This introduction requires $\mathcal{O}(\log^2 n)$ communication rounds.
The results in this chapter are published in [5], [6], [24], [40] and [23].

9.1 Computing the 2-Localized Delaunay Graph

To compute the 2-localized Delaunay graph, we use the distributed protocol described in [32]. The authors assume that an initial connected unit disk graph of all ad hoc links is given. To obtain the initial connected unit disk graph of all ad hoc links, every node executes a Wi-Fi broadcast within its transmission range. Thus, each node is aware of all nodes in its transmission range and we thus obtain a unit disk graph. The nodes use the protocol of Li et al. which requires communication costs of $\mathcal{O}(n \log n)$ bits and only $\mathcal{O}(1)$ communication rounds in the $\mathcal{LOCAL}$ model [32]. The result is not the full 2-localized Delaunay graph but instead it is a supergraph of it called *Planar Localized Delaunay Graph*. Because each edge has a length which is at most 1 and the Planar Localized Delaunay graph is a planar graph, our research ideas of hole detection also work for this type of graphs. For simplicity, we restrict ourselves to 2-localized Delaunay graphs in the rest of this chapter.

9.2 Hypercube and Convex Hull Computation

In this section, we present the protocol that transforms a ring of nodes into a hypercube structure. This protocol is a prerequisite for the convex hull calculation protocol and lets each node to detect whether it is on the boundary of a hole. The main idea of this protocol is the following: Each node of a ring chooses a predecessor and a successor. Then, it applies pointer jumping: In the first round, each node introduces its predecessor and successor to each other and a new edge between them is created. Then, the new neighbors are introduced to each other and a further edge is created and so on. For a better impression of pointer jumping, see the following figure.

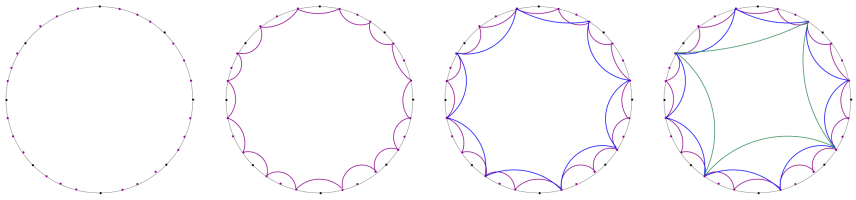

Figure 9.2: Round 0. Figure 9.3: Round 1. Figure 9.4: Round 2. Figure 9.5: Round 3.

In addition to the references of the nodes, the minimal ID of all neighbors seen so

far is exchanged in each introduction step. Eventually every node is aware of the minimal ID of all ring nodes and this node with the minimal ID becomes the leader of the ring.

This protocol is not only a prerequisite for the convex hull protocol but it also helps for a fast hole detection, i.e., it enables nodes to quickly distinguish the outer boundary from a hole. More precisely, we execute the protocol for holes and also for the outer boundary of the entire node set. Both are connected in a ring topology. The nodes of the outer boundary and the hole nodes are called *boundary nodes*. Each node v that is part of the convex hull of the entire node set detects that there are two consecutive neighbors v and w in the clockwise ordering of $v's$ neighbors with $\angle(u, v, w) \geq 180°$.
Initially, each boundary node chooses a successor and a predecessor in each ring. This works as follows: Each boundary node sorts its boundary neighbors clockwise. Then, for every pair of consecutive nodes in the sorting, included are also the last and the first node, the first node is chosen as predecessor and the second node is chosen as successor. In this way, every boundary is either oriented clockwise or counterclockwise. More precisely, the outer boundary is oriented clockwise and each hole is oriented counterclockwise. The orientation, however, is not important for the hypercube protocol but for the hole detection in Section 9.3.
To construct the hypercube we use pointer jumping. On the one hand, this technique enables us to build overlay edges for the hypercube quickly. On the other hand, it allows us to elect a leader in $\mathcal{O}(\log k)$ communication rounds and this leader is responsible for setting up the hypercube IDs. The leader of the ring is the node with minimal ID. The ID of a node v is denoted as id_v, see the following figure.

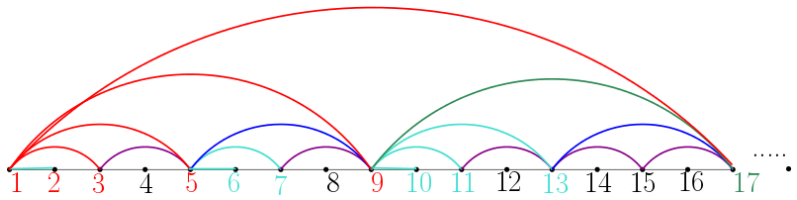

Figure 9.6: Hypercube IDs.

In addition, we assign two values to each edge $e = \{u, v\}$, which are created by the pointer jumping protocol. The first value, $\ell(e)$, defines the minimal ID of all ring nodes which are bridged by e, *except* id_u. The second value, $\text{level}(e) = \log(b)$. Here b denotes the number of ring nodes between u and v.
The detailed pointer jumping technique is used as follows: Let v be a node of the hole ring. Let $pred_0$ be its predecessor and $succ_0$ be its successor on the ring. In round 1 of the protocol, v introduces $succ_0$ to $pred_0$ to each other. In this way, $succ_0$ and $pred_0$ become adjacent nodes and an overlay edge $e = \{pred_0, succ_0\}$

is established. Further, v assigns $\ell(e) = \min\{id_v, id_{succ_0}\}$ and $\text{level}(e) = 0$. Each node executes the protocol, v also gets two nodes introduced in round 1. They are denoted as $pred_1$ and $succ_1$. In round i, each node v of the hole ring introduces its predecessor $pred_{i-1}$ to its successor $succ_{i-1}$ and gets introduced to $pred_i$ and $succ_i$. The node v that introduces $pred_{i-1}$ and $succ_{i-1}$ to each other assigns $\ell(\{pred_{i-1}, succ_{i-1}\}) = \min\{\ell(\{pred_{i-1}, v\}), \ell(\{v, succ_{i-1}\})\}$ and $\text{level}(e) = \text{level}(\{pred_{i-1}, v\}) + 1$.
With the pointer jumping technique, the hop distance between any pair of nodes halves from round to round. Finally, the protocol stops in a round i in which v gets introduced $succ_i$ and $pred_i$ and $\ell(\{pred_i, v\}) = \ell(\{v, succ_i\})$. At that point in time, each node, and thus also the leader itself, is locally aware of the minimal ID and knows the ID of the leader. Because the distance between any pair of nodes halves in each round, this protocol requires $\mathcal{O}(\log k)$ communication rounds. Being able to emulate a hypercube, we need the additional overlay edges and also hypercube IDs. Recall that the node IDs of the hypercube are bitstrings of length $\log k$. To share the hypercube IDs to the corresponding boundary nodes, the leader v assigns for each hypercube edge $\{v, succ_i\}$ the binary representation of $\text{level}(\{v, succ_i\}) + 1$ as the ID to $succ_i$. Each node receives an ID from the leader and repeats the ID distribution recursively, relative to its own ID. Because the diameter of a hypercube of k nodes is $\mathcal{O}(\log k)$, the distribution of IDs requires $\mathcal{O}(\log k)$ communication rounds. Finally, the nodes of the ring are a hypercube.

The results of this section are in the following lemma:

Lemma 9.1. A ring of k nodes can be transformed into a hypercube in $\mathcal{O}(\log k)$ communication rounds. The number of required messages is in $\mathcal{O}(\log k)$ per node.

With the hypercube, we compute the convex hull of each hole. Initially, the coordinates of points have to be sorted. Sorting n points in a hypercube is possible in $\mathcal{O}(\log n)$ communication rounds on expectation by applying the algorithm of Reif and Valiant [34]. Upon termination, Miller's algorithm can be applied. This ensures that each node of the ring knows each convex hull node. Especially each convex hull node identifies itself as a convex hull node. We obtain the following theorem:

Theorem 9.2. *Given a hole ring with k nodes, the convex hull of this hole ring can be calculated in $\mathcal{O}(\log k)$ communication rounds on expectation.*

9.3 Hole Detection

The boundary nodes cannot locally detect whether cycles are oriented clockwise or counterclockwise. Thus, they cannot detect whether they are on the outer boundary or on a hole. To let nodes distinguish the to mentioned cases they sum up angles along each boundary into the direction of the orientation. Let v_1, v_2 be a predecessor and a successor along a boundary. The case walking from v_1 to v_2 requires a left

turn and thus the angle between v_1 and v_2 is subtracted from the current sum. The angles of right turns are added. The result is 360° for the outer boundary and it is −360° for each hole [10]. The summation along a boundary can be done by a token passing technique. This technique is initiated by a leader and it requires a linear number of communication rounds for each cycle. To obtain a faster runtime, we sum up angles in parallel to the hypercube protocol: Together with the minimal ID, we exchange the sum of angles with each edge of pointer jumping. Finally, each node of the ring knows the sum of all angles along the boundary. Hence, each node decides whether it is a hole node in $\mathcal{O}(\log n)$ communication rounds. To determine the outer holes, we also need a second run of convex hull computations along the outer boundary. The outer holes are defined by an edge of the outer convex hull. When the convex hull of the outer boundary has been computed, a second run is executed between every pair of consecutive convex hull nodes. The distance exceeds the transmission range of a node.

9.4 Information Dissemination of Convex Hulls, Bounding Boxes, and Hole Rings

With the results from the previous section, each node knows if it is part of a hole or boundary or it is not. However, the holes and the hole nodes have only local knowledge. They do not have any knowledge of the existence, geographical position or the total number of holes in the entire ad hoc network. To obtain a better global knowledge of the ad hoc network such as the existence as well as the number of hole nodes, we use a broadcast mechanism to calculate an overlay that introduces them to eachother. Instead of a broadcast of all hole nodes, we use only the convex hulls and the bounding boxes as hole abstractions. Hence, we still need to distribute the information about convex hulls and bounding boxes in the entire network.

We use an overlay network via long-range links. This overlay has only a logarithmic diameter [16]. The protocol makes sure that all nodes of the network are connected in a rooted tree via long-range links after $\mathcal{O}(\log^2 n)$ communication rounds. This tree has a height of $\mathcal{O}(\log n)$ and a constant degree and hence the diameter of the tree is $\mathcal{O}(\log n)$. The tree lets us distribute references of all convex hull nodes in $\mathcal{O}(\log n)$ communication rounds, i.e., each convex hull node can direct its own reference towards the root and into the subtree below itself. The root sends the reference into every other subtree. This protocol avoids that nodes receive the same broadcast message twice ore more often. The total runtime of it is $\mathcal{O}(\log^2 n)$ because the tree has to be established initially.

With the above protocol of pointer jumping, we can build hypercube edges. The node IDs of the hypercube are bitstrings of length $\mathcal{O}(\log k)$. To broadcast the hypercube IDs to the corresponding hole node, we select a leader of the hole node ring which is min_{id}. After the leader election of min_{id}, min_{id} broadcasts the hypercube

IDs to the corresponding ring node. This takes $\mathcal{O}(\log k)$ rounds.
Sorting all ring hole nodes $\{r_1, ..., r_k\}$ such that the x-coordinate of r_1 is less than the x-coordinate of r_k is done with Batcher's Bitonic Sort on the hypercube of the hole ring with parallel runtime $\mathcal{O}(\log^2 n)$ Finally, we now apply the parallel convex hull algorithm by [34]. This takes in $\mathcal{O}(\log n)$ rounds.

Theorem 9.3. *Given a hole ring with k nodes, the convex hull of this hole ring can be calculated in $\mathcal{O}(\log^2 k)$ rounds.*

The construction of overlay edges causes extra costs and is described by the following corollary.

Corollary 9.4. The message complexity for computing a convex hull of a hole ring with k nodes is $\mathcal{O}(k \log k)$.

Proof: Each nodes receives $\mathcal{O}(\log k)$ messages while Batcher's bitonic sort and the parallel convex hull algorithm of Miller are applied. This results in a total message complexity of $\mathcal{O}(k \log k)$. Further, the construction of the hypercube has the same message complexity. Because each node of a hole ring introduces two of its neighbors to each other in every round, a node sends two messages per round. Because the number of rounds is $\mathcal{O}(\log k)$, the total costs are $\mathcal{O}(k \log k)$. □

Let us analyze the storage demands for each node lying on the perimeter of a hole.

Corollary 9.5. For computing a convex hull of k nodes, $\mathcal{O}(\log k)$ references have to be stored at each node.

Proof: A hypercube has logarithmic degree. Thus, each node has to store $\mathcal{O}(\log k)$ references to its hypercube neighbors. The pointer jumping technique for computing the hypercube does not require more storage, because each node gets introduced two neighbors per round.

For the computation of a convex hull, Batcher's bitonic sort is applied, This does not incur more costs, because each node compares values of hypercube neighbors per round. Miller's algorithm needs $\mathcal{O}(\log k)$ storage per node [34]. Because each node has at most two convex hull neighbors, the total space requirements are $\mathcal{O}(\log k)$. □

Overlay Structure: After the computation of convex hulls, we are interested in connecting all convex hulls in an overlay structure. This overlay is a clique of all convex hull nodes. Because a convex hull node can locally not decide in which direction there are other convex hull nodes, we use broadcast mechanisms to build a clique of all convex hull nodes. Because a broadcast needs at least as many rounds as the diameter of the network, this takes up to $\mathcal{O}(n)$ communication rounds. Now,

each convex hull node computes a Delaunay graph of all convex hull nodes locally. To do so, the algorithm of Guibas et al. can be applied [13].
We need a another broadcast mechanism to make sure that each node of the ad hoc network knows its closest convex hull node such that shortest path-queries are directed to these nodes. Here, we use local broadcasts and not every broadcast is flooded through the whole ad hoc network. This takes up to $\mathcal{O}(n)$ rounds. Instead of this, a node that receives two broadcast messages decides which of these message has to be forwarded to which neighbor, because each node knows the geographical positions of its neighbors.

Finally, nodes of the ad hoc network can send shortest path queries to their closest convex hull nodes. The query for a shortest path are the coordinates of a source node s and a target node t. The convex hull node that receives this query inserts s and t into the Delaunay graph of convex hull nodes and compute a shortest path between s and t afterwards. The integration into the Delaunay graph takes constant time in expectation with the help the algorithm of Guibas et al. [13]. The shortest path is computed with Dijkstra's algorithm using Fibonacci Heaps with runtime $\mathcal{O}(|V| \log |V| + |E|)$. We have a runtime of $\mathcal{O}(k \log k)$, where k denotes the number of all convex hull nodes, because $E \in \mathcal{O}(V)$ for planar graphs.

Bounding Boxes. The hypercube is also used for a fast dissemination of bounding box information. The nodes of the same hole exchange their coordinates and keep the maximal and minimal value of both x- and y-coordinates. After $\mathcal{O}(\log n)$ communication rounds, each node knows of the bounding box coordinates of its hole. The node with the smallest x-coordinate is the responsible node for exchanging bounding box information with other bounding boxes. To do this, an overlay tree according to the protocol in [16] is built initially. Also here, the initial setup of the tree requires $\mathcal{O}(\log^2 n)$ communication rounds. The node with the smallest x-coordinate sends the bounding box information of its hole up and down in the tree. The representatives of a bounding box do the computation of c-competitive paths in the ad hoc network and they store the modified bounding box visibility graph locally. All the other nodes of the network only store whether they are located on the boundary of a bounding box or not.

10 Efficient SetCover

In this chapter, we present a distributed algorithm to solve SetCover. This algorithm works in the KT_0 model. For the definition of the KT_0 model, see Chapter 2. The results of this chapter are published in [18].

In Chapter 4, we mentioned a distributed algorithm for SetCover that works in the Beeping model. In this chapter, we move away from this model and present a low-message KT_0-Congest algorithm to solve this problem. The KT_0-Congest model is a special type of the synchronous Congest model. Here, the computation happens in discrete (synchronous) steps. In each step, a logarithmic number of bits (in the network size) can be exchanged per edge. In addition, each node has a unique identifier. The message complexity of a lot of distributed algorithms depends on the initial knowledge of the nodes. It is important that the nodes know the identifiers of their neighbors in the communication graph ([17] gives an overview). There are two models. In the KT_0 (i.e., **K**nowledge **T**ill radius **0**) only the node's own identifier is known. In the KT_1 (i.e., **K**nowledge **T**ill radius 1) model even neighbors' identifiers are known. In this thesis, we consider the KT_0 model, i.e., nodes only know their own identifier. Further, the nodes are (initially) oblivious of all other identifiers. In some literature, the KT_0 model is called the *clean network model* [36]. This model does not represent ad-hoc networks faithfully. Instead it is used for the analysis of the message efficiency of distributed algorithms. Recall that we make the following less restrictive assumptions about the KT_0 model:

1. We consider a fixed communication graph $G := (V_S \cup V_{\mathscr{U}}, E)$ with $V_S = n$ and $V_{\mathscr{U}} = m$ with bidirected communication edges $\{s, e\} \in E$ between a set and its elements. Sets and elements are able to locally distinguish between their edges through port numbers. There are no global identifiers uniquely who identify a node. All nodes know Δ and $\log \tilde{n}$ to simplify the presentation.

2. Time proceeds in synchronous *slots*[*] as in the Beeping model. Different from the Beeping model, an element or set is able to send a distinct message of size $O(\log(n+m))$ to any subset of its neighbors. Each element and set can receive a distinct message from each of their neighbors. All messages are received in the next slot.

3. All nodes wake up in the same slot and start the algorithm synchronously.

[*] In some litereature, the term *round* is used to describe a communication step. Instead,we use the term slot throughout this article. First, the term *round* was used in describing the Beeping algorithm. Second, our goal is to have a consistent term for a communication step.

Recall that we want to find a solution set such that a) each element has a neighbor in this set and b) this set is as small as possible. As in the BEEPING model, each set and element starts with limited knowledge of its neighborhood and has to *learn* everything it needs to solve the given problem. We are interested in the number of messages and we want to prove that our algorithm sends a sublinear (in the number of edges) number of messages, w.h.p. Note that *w.h.p.* means: The probability of failure is polynomially small in the input size. In more detail, the input size is $n + m$ which is the sum of both sets and elements. We are searching for failure probabilities in the magnitude of $o\left((n+m)^{-c}\right)$ for a tunable $c > 1$. We define $\tilde{n} := n + m$.

Under the above mentioned assumptions, we will show the following theorem:

Theorem 10.1. *There is an algorithm in the KT_0-CONGEST model that solves* SETCOVER *in time $O(k^2)$, expected approximation ratio of $O(\Delta^{\frac{1}{k}} \log \Delta)$, and sends only $\tilde{O}\left(\Delta^{\left(\frac{1}{2}+\frac{1}{k}\right)}(n+m)\right)$ messages, w.h.p., given that all nodes know Δ, k, and $\log \tilde{n}$.*

For $k \in \Theta(\log \Delta)$ we present an algorithm with polylogarithmic runtime and approximation ratio. This algorithm uses $O(\sqrt{\Delta}n)$ messages, see Section 10.1 for the algorithm and Section 10.2 for the analysis of the algorithm.

Furthermore, we present a lower bound on the number of messages that are needed to approximate a solution. This lower bound holds for the KT_0 algorithms that want to achieve a non-trivial approximation ratio. The approximation ratio is part of any lower bound for the problem. A trivial algorithm just adds all sets to the solution and sends no messages. In this case, the approximation ratio is $O(\Delta)$ and this can be up to $O(m)$. We will show that:

Theorem 10.2 (Lower Bounds)**.** *Let $\alpha \in [1, \log n]$ be an approximation factor. Then, there are instances for* SETCOVER*, such that:*

1. *Every randomized KT_0-CONGEST algorithm that yields an α-approximation with probability $\geq \frac{2}{3}$ needs at least $\Omega\left(m\Delta\right)$ messages for $\alpha < 1.01$.*
2. *Every randomized KT_0-CONGEST algorithm that yields an α-approximation with probability $\geq \frac{2}{3}$ needs at least $\Omega\left(m\Delta^{\frac{1}{2\alpha}}\right)$ messages for any $\alpha \geq 1.01$.*

Our algorithm does not exactly match these lower bounds but they still have major implications for our result. First, every distributed algorithm that aims for an exact solution has to send a lot of messages. Which means that every message-efficient algorithm can only produce an approximation even if with unlimited local computational power within the nodes. This justifies that our algorithm calculates an approximation for the solution, because we do not want to hope for an exact solution that needs a sublinear amount of messages. We also get a message bound for a given approximation ratio α. Compared to this bound, the performance of our algorithm is rather mediocre. For a constant approximation factor $\alpha = 1 + \epsilon$ with

$\epsilon \geq 0.1$, the bound is $\Omega(m\Delta^{\frac{1}{2}-\epsilon'})$ for some small $\epsilon' > 0$. By choosing $k \in O(\log \tilde{n})$, our algorithm has the message complexity of $O((m+n)\sqrt{\Delta})$. It comes reasonably to this bound[†]. With $k \in O(\log \tilde{n})$, our algorithm has a logarithmic approximation ratio on general graphs. This is worse than the constant factor $\alpha = 1 + \epsilon$. But one could look at this fact from a different perspective and see that the lower bound for a logarithmic approximation ratio is $\Omega(n)$ which means only one message per element. Our algorithm produces results that are far away from this for graphs of high degree. So there is still room left for improvement. Nevertheless, we the bound that we present is useful as a benchmark for (possible) future algorithms. Please find the bounds in Section 10.3. They are based on a result by Indyk et al. [20] that is given for a stronger sequential model.

10.1 The Algorithm

In this section, we present the algorithm behind Theorem 10.1. We only need to make a couple of changes to the Beeping-Algorithm to prove this theorem. Every Beeping-Algorithm works in the KT_0-Congest-Model because the model is less restricted. A straightforward simulation of a Beeping algorithm would let all nodes send a message to all their neighbors and would not exploit the stronger model. This could lead to a lot of superfluous messages. Since our goal is to reduce the number of messages, we must be more careful. We only need to make a few changes to the Beeping algorithm to obtain a message-efficient algorithm. These changes mainly affect the way in which the uncovered elements are counted. The algorithm has k phases and $4k+1$ rounds in each phase. Each round only takes 2 slots. At the beginning of each phase, all sets pick a truncated geometric random variable $X_s \in [0, 4k]$ with parameter $1 - \Delta^{-\frac{1}{k}}$ and wait until round $\Phi_s = 4k - X_s$ try and enter the solution. Thus, this part of the algorithm is the same as before. The algorithm works differently for the elements as we want to count them more efficiently. In the this algorithm, instead of sampling $4k$ variables $Y_e^1, \ldots, Y_e^{4k}$, the elements perform a random experiment for each edge. In more detail, for each incident edge (e, s), an element $e \in V_{\mathscr{U}}$ draws $c \cdot 4 \cdot \log \tilde{n}$ binary random variables $Y_{(e,s)}^1, \ldots, Y_{(e,s)}^{c4 \log \tilde{n}}$ uniformly and independently at random. The variable $c > 6$ is a security parameter. This parameter can trade message complexity for success probability. In phase i, we have $Y_{(e,s)}^{\ell} = 1 = \frac{1}{\Delta_i}$ as in the previous algorithm. These variables determine the number of messages that will be sent along the edge. All variables are drawn at the beginning of each phase and the same random variables are used in every round of that phase.

Given these variables, a single round j in phase i works as follows: Instead of beeping *all* neighbors in certain slots, an uncovered element e sends $Y_{(e,s)} := \sum_{\ell=1}^{c4 \log \tilde{n}} Y_{(e,s)}^{\ell}$ Beeps to set $s \in N_e$. After this, the sets count how many Beeps they received. This is not possible in the Beeping algorithm because we cannot

[†]Note: With lower k, the approximation factor and the message complexity rise. Thus a logarithmic k is the best choice.

distinguish between the origin of different BEEPs and cannot have more than one BEEP per edge. Thus, this allows for a more precise approximation of the elements. If a set receives more than $c \log \tilde{n}$ BEEPs and the set holds $\Phi_s = j$, it enters the solution and tells its neighbors about it by sending a message to all of its neighbours. This can be done in 2 slots of sending and receiving. Thus, a round only needs two slots.

The mentioned changes above are not enough to obtain a message-efficient algorithm. For example, the sets that join the solution, still need to inform all of their neighbors once they enter the solution. If some of their neighbors are covered, they receive messages that could have been avoided. In addition, we cannot let the covered elements inform their corresponding sets. Each element is covered eventually and this would be doomed to send $\Omega(\Delta n)$ messages in the worst case. Thus, the elements should only do this at certain points in time.

To manage all of this, we make the following change to the algorithm that provides a trade-off. First, we execute our algorithm until phase $\frac{k}{2}$. We will see, at most $\tilde{O}\left(\Delta^{\frac{1}{k}} n\sqrt{\Delta}\right)$ sets joined the solution *and* all remaining sets cover around $O(\sqrt{\Delta})$ elements at this point, w.h.p. This is called the first *stage* of the algorithm. Denote the remaining uncovered elements at this point as $\mathscr{U}'$. After the first stage ,and only then, all the uncovered elements tell their neighboring sets that they are uncovered. To do so, all elements in $V'_{\mathscr{U}}$ send a message to all their neighboring sets. Therefore, the number of edges between uncovered elements and sets bounds the number of messages. With the argument from above, this is at most $O(\sqrt{\Delta}m)$, w.h.p. The algorithm continues almost as usual for the remaining $\frac{k}{2}$ phases. But each set that joins the solution notifies *only* the elements in $V'_{\mathscr{U}}$ that they are covered and does not send additional messages to the already covered elements. Finally, in each phase, $\tilde{O}(\sqrt{\Delta}m)$ messages are sent. This is called the second *stage* of the algorithm.

The pseudocode is shown in Figure 10.1. This code consists of two procedures for each type of node. First, there are the procedures SETS and ELEMENTS. They correspond to the eponymous procedures in Figure 4.4 and they obtain the changes mentioned above. The new parameters u and l specify which phases should be executed. This means that the algorithm executes the phases from u to l. Further, the security parameter $c > 6$ is given to the algorithm. Finally, both sets and elements have an outer loop. This loop handles the division into the two stages mentioned above. The procedure OUTERLOOP executes the algorithm with the appropriate parameters.

10.2 Analysis of the Algorithm

In this section, we prove Theorem 10.1 and we prove that our algorithm has the above mentioned runtime, approximation ratio, and message complexity. The runtime of the algorithm is $O(k^2)$ because the algorithm only consists of deterministic loops. But what about the approximation ratio? It is easy to see that most of the lemmas from the analysis in Chapter 4 are still correct. More concrete, we use the

Algorithm 5 BEEP-AND-SLEEP: CODE FOR SETS

procedure SETS($s, N_s, k, \Delta, u, l, c$)
  $S_s \longleftarrow 0$
  for $i := u, \ldots, l$ **do**
    Pick $X_s \sim Geo(1 - 1\Delta^{1k})$.
    $X_s \longleftarrow \min\{X_s, 4k\}$

    for $j := 0, \ldots, 4k$ **do**

      $A \longleftarrow \{$BEEPS *received* $\}$

      if $(|A| \geq c \cdot \log \tilde{n}) \wedge (4k - X_s = j)$ **then**
        Send BEEP to all $e \in N_s$.
        $S_s \longleftarrow 1$
      end if
    end for
  end for
  return S_s
end procedure

procedure OUTERLOOP(s, N_s, k, Δ, c)
  $S_s \longleftarrow$ SETS($s, N_s, k, \Delta, \Delta, \sqrt{\Delta}$)
  if $S_s = 0$ **then**
    $N'_s \longleftarrow \{e \in N_s \mid e$ *sent* BEEP$\}$

    $S_s \longleftarrow$ SETS($s, N'_s, k, \Delta, \sqrt{\Delta}, 0$)
  end if
end procedure

Algorithm 6 BEEP-AND-SLEEP: CODE FOR ELEMENTS

procedure ELEMENTS($e, N_e, k, \Delta, u, l, c$)
  $C_e \longleftarrow 0$
  for $i := u, \ldots, l$ **do**
    for $s \in N_e$ **do**
      for $\ell \in 1, \ldots, c \cdot 4 \cdot \log \tilde{n}$ **do**
        $Y^{\ell}_{(e,s)} \sim B\left(\frac{1}{\Delta_i}\right)$
      end for
    end for

    for $j := 0, \ldots, 4k$ **do**
      for $s \in N_e$ **do**
        $Y_{(e,s)} \longleftarrow \sum_{\ell=1}^{c \cdot 4 \cdot \log \tilde{n}} Y^{\ell}_{(e,s)}$
        if $(Y_{(e,s)} \geq 1) \wedge (C_e = 0)$ **then**
          Send $Y_{(e,s)}$ BEEPs to s.
        end if
      end for

      $l \longleftarrow \{s \in N_e \mid s$ *sent* BEEP$\}$
      If $l \neq \perp$, set $C_e \longleftarrow 1$

    end for
  end for
  return C_e
end procedure

procedure OUTERLOOP(s, N_s, k, Δ, c)
  $C_e \longleftarrow$ SETS($s, N_s, k, \Delta, \Delta, \sqrt{\Delta}$)
  if $C_e \neq 0$ **then**
    Send BEEP to all $s \in N_e$
  else
    $C_e \longleftarrow$ SETS($s, N_e, k, \Delta, \sqrt{\Delta}, 0$)
  end if
end procedure

Figure 10.1: The pseudocode depicts the code for the elements (right) and the code for the sets (left). Sets and elements are synchronized. Whenever the sets send a message, the elements receive a message and vice versa.

observation that two factors mainly determine the quality of the solution:

1. The number of sets that add themselves simultaneously when an element $u \in V_{\mathscr{U}}$ is covered.

2. The ratio between the best set's span and the span of the set covering u.

These values are denoted as $\mu_{(i,j)}(u)$ and $\eta_{(i,j)}(u)$ respectively. The way of how the sets add themselves is the same. The proof of Lemma 4.18 can be used for the new counting mechanism[‡]. Thus, $\mu_{(i,j)}(u)$ can be bounded as before. There are some differences in the analysis of the $\eta_{(i,j)}(u)$ because we are able to count more precisely. Note that in the proof of Lemma 4.19, the main idea was to bound two random values. There was the maximal number of uncovered elements of an element in phase i which is bounded in Claim 4.5. The minimal number of uncovered elements was considered of any set that joins the solution in Claim 4.5. It suffices to recreate these two bounds for our adapted analysis with the following lemma:

Lemma 10.3. The following two statements hold w.h.p:

1. At the end of phase i there is no set with Δ_i uncovered elements.

2. Any set that adds itself in phase i covers at least $\Delta_i 8$ elements.

Let s be a set with uncovered element on wake-up. Let $\mathscr{Y}_s^i$ denote the number of BEEPS it receives. For all its edges to uncovered nodes, s independently receives $c \cdot 8 \cdot \log \tilde{n}$ BEEPs with probability $\frac{1}{\Delta_i}$ each. Let the number of BEEPs that u sends to s (if it is uncovered) be $Y_{(u,s)} = \sum_{\ell=1}^{c4 \log \tilde{n}} Y_{(u,s)}^{\ell}$. We prove both statements separately:

1. Let s be a set with Δ_i uncovered elements at the end of phase i. Then, these uncovered elements were neither covered by s nor any other set containing them. We ignore the other sets in this proof. We only focus on s. This upper bounds the sought-after probability because we ignore events that reduce the number of uncovered elements. We upper bound the probability that s adds itself in phase i. Note that all sets will add themselves in phase i. They cover all their neighbors if and only if they receive more than $c \cdot 4 \cdot \log \tilde{n}$ BEEPs when they wake up. Let us consider the round (of phase i) in which s wakes up. Since the number of its uncovered neighbors can only decrease, s has at least Δ_i uncovered elements in this round. Thus, the expected number of BEEPs

[‡]Only Equation (4.49) changes as we need to use $Y_{(u,s)}^{\ell}$ instead of Y_u^{ℓ}. The rest is the same.

is:

$$E[\mathscr{Y}_s^i] = E\left[\sum_{u \in N_s, C_u = 0} Y_{(u,s)}\right] = \sum_{u \in N_s, C_u = 0} E[Y_{(u,s)}] \tag{10.1}$$

$$= \sum_{u \in N_s, C_u = 0} c \cdot 8 \cdot \log \tilde{n} \cdot \frac{1}{\Delta_i} \tag{10.2}$$

$$= \Delta_i \cdot c \cdot 8 \cdot \log \tilde{n} \cdot \frac{1}{\Delta_i} \tag{10.3}$$

$$= c \cdot 8 \cdot \log \tilde{n} \tag{10.4}$$

All uncovered elements *indepedently* send a BEEP to s. Thus, the variable

$$\mathscr{Y}_s^i := \sum_{u \in N_s} Y_{(u,s)} = \sum_{u \in N_s, C_u = 0} \sum_{\ell=1}^{c4 \log \tilde{n}} Y_{(u,s)}^{\ell} \tag{10.5}$$

is the sum of independent binary random variables. By applying the Chernoff bound, the probability that only $c \cdot 4 \cdot \log \tilde{n}$ BEEPS are sent is:

$$\mathscr{Y}_s^i \leq c \cdot 4 \cdot \log \tilde{n} \leq A_s^i \leq \left(1 - \frac{1}{2}\right) E[\mathscr{Y}_s^i] \tag{10.6}$$

$$\leq e^{\frac{c8 \log \tilde{n}}{2 \cdot 2^2}} = \frac{1}{\tilde{n}^c} \tag{10.7}$$

With more elements that are uncovered, the probability can only be smaller. Thus, the probability that s is not added on wake-up and therefore still has at least Δ_i uncovered neighbors at the end of the phase is smaller than $1\tilde{n}^c$.

2. Let us consider a set $s \in V_S$ that adds itself to the solution in phase i. Note that there is precisely one well-defined round in which s can add itself. This round is its wake-up round $\Phi_s := 4k - X_s$. In addition, s will only add itself if it receives a BEEP from $c \cdot 4 \cdot \log \tilde{n}$ elements. We will prove that this does not happen w.h.p. Hence, to bound the probability that s has $\frac{\Delta_i}{8}$ uncovered elements on joining the solution, we assume that s has $\frac{\Delta_i}{8}$ uncovered elements on wake-up and we bound the number of BEEPS. To do so, we first compute the expected values. Those are:

$$E[\mathscr{Y}_s^i] = E\left[\sum_{u \in N_s, C_u = 0} Y_{(u,s)}\right] = \sum_{u \in N_s, C_u = 0} E[Y_{(u,s)}] \tag{10.8}$$

$$\leq \frac{\Delta_i}{8} \cdot c \cdot 8 \cdot \log \tilde{n} \cdot \frac{1}{\Delta_i} \tag{10.9}$$

$$= c \log \tilde{n} \tag{10.10}$$

As in the previous case, the variable $\mathscr{Y}_s^i := \sum_{u \in N_s} Y_{(u,s)}$ is the sum of independent binary random variables. We can apply the Chernoff bound. This

time, we apply it to obtain an upper bound on the probability. Thus, the probability that at least $c \cdot 4 \cdot \log \tilde{n}$ BEEPs are sent is:

$$\mathscr{Y}_s^i \geq c \cdot 4 \cdot \log \tilde{n} \leq A_s^i \geq (1+3)E[\mathscr{Y}_s^i] \tag{10.11}$$

$$\leq e^{\frac{3c \log \tilde{n}}{3}} \leq \frac{1}{n^c} \tag{10.12}$$

The probability can only be smaller if s fewer uncovered elements. Hence, the lemma follows.

In phase i, all sets that enter the solution cover between $\frac{\Delta_i}{8}$ and Δ_{i-1} uncovered elements w.h.p. This directly bounds the difference between the *worst* and *best* set that covers a given element to $O(\Delta^{\frac{1}{k}})$. Given that Lemma 4.21 still holds, we can further conclude that the expected approximation ratio is $O(\Delta^{\frac{2}{k}} \log \Delta)$. This is slightly better than our BEEPING algorithm.

We analyze the message complexity and we prove the message bound for each stage of the algorithm. We begin with the first stage and show that:

Lemma 10.4. Until phase $\frac{k}{2}$, the nodes send at most $\tilde{O}(\Delta^{\frac{1}{2}+\frac{1}{k}} n)$ messages w.h.p.

We prove the lemma separately for the elements and sets. We show that either class of nodes sends at most $\tilde{O}(\Delta^{\frac{1}{2}+\frac{1}{k}} n)$ messages, w.h.p.

1. Fix a phase $i \leq \frac{k}{2}$. Elements only send messages to indicate whether they are uncovered or not. We refer to an edge (e, s) via which we send at least one BEEP in phase i as an *active* edge. Let $A_s^i(u)$ denote the event that e has an active edge to s. We send at most $O(\log \tilde{n})$ BEEPs along an active edge. These can be encoded in $O(\log \log \tilde{n})$ bits and use only a single message. Thus, at the beginning of each phase, an element picks a set of active edges and sends exactly one message along each edge in each round of the phase. Hence, it suffices to bound $A_s^i(u)$ for all phases $i \leq \frac{k}{2}$ and then multiply it by $4 + 1k$, the number of rounds per phase. In phase $i \leq \frac{k}{2}$, each element picks an edge to some set $s \in N_u$ with the probability:

$$Pr[A_s^i(u)] = Pr[Y_{(e,s)} \geq 1] \tag{10.13}$$

$$= Pr[\bigcup_{\ell=1}^{c \cdot 4 \cdot \log \tilde{n}} Y_{(e,s)}^{\ell} = 1] \leq \sum_{\ell=1}^{c \cdot 4 \cdot \log \tilde{n}} Pr[Y_{(e,s)} = 1] \quad \triangleright \textit{Union bound} \tag{10.14}$$

$$\leq c \cdot 8 \cdot \log \tilde{n} \cdot \frac{1}{\Delta_i} = c \cdot 8 \cdot \log \tilde{n} \cdot \frac{\Delta^{\frac{i}{k}}}{\Delta} \tag{10.15}$$

$$\leq c \cdot 8 \cdot \log \tilde{n} \cdot \frac{\Delta^{\frac{1}{2}}}{\Delta} \quad \triangleright \textit{As } i \leq k/2 \tag{10.16}$$

$$= c \cdot 8 \cdot \log \tilde{n} \cdot \frac{1}{\sqrt{\Delta}} \tag{10.17}$$

Thus, the expected number of active edges is:

$$E\left[\sum_{s\in N_u} A_s^i(u)\right] := \sum_{s\in N_u} E[A_s^i(u)] \leq \sum_{s\in N_u} c\cdot 8\cdot \log\tilde{n}\cdot\frac{1}{\sqrt{\Delta}} \leq \Delta\cdot c\cdot 8\cdot\log\tilde{n}\cdot\frac{1}{\sqrt{\Delta}} \tag{10.18}$$

$$\leq c\cdot 8\cdot\log\tilde{n}\cdot\sqrt{\Delta} \tag{10.19}$$

Since the edges are picked independently, they can be subjected to the Chernoff bound. Thus, we assume that all elements pick at most $\tilde{O}(\sqrt{\Delta})$ edges per phase, w.h.p. Recall that a node sends at most one message per round over each active edge. Hence, each element sends at most $\tilde{O}\left(k\sqrt{\Delta}\right)$ messages in each phase. Because we only consider the first $\frac{k}{2}$ phases, the total number messages is $\tilde{O}\left(k^2\sqrt{\Delta}\right)$. We assume that $k \in O(\log n)$ because higher values of k do not improve the approximation ratio, each element sends $\tilde{O}(\sqrt{\Delta})$ messages as $k^2 \in O(\log^2 n)$ disappears in the $\tilde{O}$-notation. The sum of all elements yields the statement.

2. Sets send messages whenever they add themselves to the solution. More concrete, each set that adds itself sends at most Δ messages to let this know to all its neighboring elements and remains silent otherwise. Hence, we show that at most $\tilde{O}\left((\Delta^{\frac{1}{k}}\sqrt{\Delta})n\right)$ sets add themselves w.h.p. First, note that every set that adds itself covers at least $\sqrt{\Delta}/8$ uncovered elements, w.h.p, otherwise it would *not* have added itself. This comes from the second statement in Lemma 10.3. The meaning of the statement is that each set that adds itself in phase i covers $\frac{\Delta_i}{8}$ elements, w.h.p. Using that $i \leq \frac{k}{2}$, we see:

$$\frac{\Delta_i}{8} := \frac{\Delta^{\frac{k-i}{k}}}{8} \geq \frac{\Delta^{\frac{k-\frac{k}{2}}{k}}}{8} = \frac{\sqrt{\Delta}}{8} \tag{10.20}$$

We apply the union bound and we conclude that all sets cover at least $\frac{\Delta_i}{8}$ elements, w.h.p. Let $S_i \subset V_S$ denote the sets that add themselves in phase i. To prove the statements, we must show that at most $\Delta^{\frac{1}{k}}n$ sets add themselves where n is the number of elements and not sets. This follows from the algorithm's approximation ratio, which states that no element can be covered by too many sets. More concrete, we compute the number of sets that cover each element to get a precise bound. To do so, we claim that each uncovered element is covered by *at most* $O(\Delta^{\frac{1}{k}}\log\tilde{n})$ sets, w.h.p. This follows from Lemma 4.21. Fix an element e covered in phase i and let μ_e be the number of sets simultaneously sending a message to it. Then, by Lemma 4.21, we have:

$$Pr[\mu_e > t] \leq \max\left(e^{-t/4}, 2(1-\Delta^{-\frac{1}{k}})^{t-1}\right) \tag{10.21}$$

Choosing $t \geq 4\Delta^{\frac{1}{k}} c \log \tilde{n} + 1$ for some $c > 0$, it holds:

$$Pr[\mu_e > 4\Delta^{\frac{1}{k}} c \log \tilde{n}] \leq \max\left(e^{-\Delta^{\frac{1}{k}} c \log \tilde{n} + \frac{1}{4}}, 2(1 - \Delta^{-\frac{1}{k}})^{4\Delta^{\frac{1}{k}} c \log \tilde{n}}\right) \tag{10.22}$$

$$\leq \max\left(\frac{1}{\tilde{n}^c}, 2e^{-4c \log \tilde{n}}\right) \tag{10.23}$$

$$\leq \max\left(\frac{1}{\tilde{n}^c}, e^{-c \log \tilde{n}}\right) \tag{10.24}$$

$$\leq \frac{1}{\tilde{n}^c} \tag{10.25}$$

Hence, by applying the union bound, every element that is covered in this phase is covered by $t \in O(\Delta^{\frac{1}{k}} \log \tilde{n})$ sets, w.h.p. Let C_i be the covered elements and let M_i be the set of all messages exchanged between sets and *uncovered* elements in phase i. As each set $e \in C_i$ receives at most $4\Delta^{\frac{1}{k}} c \log \tilde{n} + 1$ messages, we have:

$$M_i \leq 4\Delta^{\frac{1}{k}} c \log \tilde{n} + 1 \tag{10.26}$$

Further, by our argument from above, it holds:

$$|S_i| \cdot \frac{\sqrt{\Delta}}{8} \leq M_i \tag{10.27}$$

Then, by combining these two observations, we see that it must hold:

$$|S_i| \cdot \sqrt{\Delta}/8 \leq 4 \cdot c \cdot \Delta^{\frac{1}{k}} \cdot \log \tilde{n} \cdot |C_i| + 1 \tag{10.28}$$

$$\Leftrightarrow |S_i| \leq \frac{32 \cdot c \cdot \Delta^{\frac{1}{k}} \cdot \log \tilde{n} \cdot |C_i|}{\sqrt{\Delta}} + 8 \tag{10.29}$$

$$\Leftrightarrow |S_i| \leq \frac{33 \cdot c \cdot \Delta^{\frac{1}{k}} \cdot \log \tilde{n} \cdot |C_i|}{\sqrt{\Delta}} \tag{10.30}$$

$$\Leftrightarrow |S_i| \leq \frac{33 \cdot c \cdot \Delta^{\frac{1}{k}} \cdot \log \tilde{n}}{\sqrt{\Delta}} n \tag{10.31}$$

$$\Leftrightarrow |S_i| \in \tilde{O}\left(\frac{\Delta^{\frac{1}{k}}}{\sqrt{\Delta}} n\right) \tag{10.32}$$

If there are more than $\tilde{O}\left((\Delta^{\frac{1}{k}}\sqrt{\Delta})n\right)$ sets that added themselves, then there must exist an element covered by more than $4c\Delta^{\frac{1}{k}} \log \tilde{n} + 1$ sets. We assume that fewer sets cover each element, so this is a contradiction.

We combine the findings to prove the statement. First, recall that all sets send at most $\Delta|S_i|$ messages in phase i. If all sets cover more than $\sqrt{\Delta}/8$ uncovered element *and* all uncovered elements are covered by at most $3c\Delta^{\frac{1}{k}} \log n + 1$ sets,

we have $|S_i| \in \tilde{O}\left(\Delta^{\frac{1}{k}-\frac{1}{2}}n\right)$ by the argument above. Since, these two events hold w.h.p., it follows that we also send $\tilde{O}\left(\Delta^{\frac{1}{2}+\frac{1}{k}}n\right)$ messages w.h.p. A union bound over all $\frac{k}{2}$ phases yields that $\tilde{O}\left(k\Delta^{\frac{1}{2}+\frac{1}{k}}n\right)$ messages are sent in total, w.h.p. As $k \in O(\log(\tilde{n}))$, we obtain the lemma.

This lemma proves the analysis of the algorithm's first stage. We also need to consider the second stage. In this stage, the algorithm uses communication edges adjacent to the uncovered elements in phase $\log \Delta/2$. Hence, we need to count these edges to determine the message complexity. We prove:

Lemma 10.5. At the end of phase $\frac{k}{2}$, each set has at most $O(\sqrt{\Delta})$ uncovered elements w.h.p.

This lemma follows from the first statement of Lemma 10.3 because all sets that have more than $O(\sqrt{\Delta})$ uncovered elements must have added themselves in an earlier round, w.h.p. Hence, in the remainder of the algorithm, all communication will take place via these $\tilde{O}\left(m\sqrt{\Delta}\right)$ edges. Because at most one message passes each edge in every slot and the runtime is $O(k^2) \subseteq \tilde{O}(1)$, we can imply that at most $\tilde{O}\left(m\sqrt{\Delta}\right)$ messages are sent in the second stage. Combining this with the fact that $O(\Delta^{\frac{1}{2}+\frac{1}{k}}n)$ messages are sent in the first stage proves the main theorem of this chapter.

10.3 Lower Bound and upper Bound

In this section, we prove Theorem 10.2. The proof is a reduction to the sequential model presented in [20]. It does not consider message passing but instead makes queries to a SetCover instance stored in memory. More concrete, they consider the following two queries:

- `EltOf(i,j)` - Returns the j^{th} element of Set S_i or $\perp$ if there is no such element.
- `SetOf(i,j)` - Returns the j^{th} set which contrains u_i or $\perp$ if there is no such set.

In this model, every randomized approximation algorithm must query a large portion of the input. This means that the connections between the nodes must be revealed to the algorithm. This is *not* a distributed model. Thus, no messages are sent. Hence, one can perform computations on *all* elements queried from the input. This computational power only helps if the algorithm queries enough sets and elements, see [20] Before we go into the details of their lower bound, we first show that the sequential model is indeed more powerful than our KT_0 model. More concrete, we prove that a distributed algorithm that sends less than x messages can

be turned into a sequential algorithm with less than x queries. Thus, any lower bound on the queries is also a lower bound for messages. The proof's main ingredient is the observation that every message that is sent from an element $e \in V_{\mathscr{U}}$ along its j^{th} channel can be emulated as looking up `SetOf(e,j)`. The same holds also for a set $s \in V_S$ with `EltOf(s,j)`. We obtain:

Lemma 10.6. Any algorithm that yields an α-approximation for SetCover in the KT_0 model that sends x messages can be transformed into a sequential algorithm that makes x queries.

First, we create $m + n$ objects. The objects store the internal variables of each set and element. They are stored in two arrays, A_S and A_E, s.t., the object for element u_i can be accessed through $A_E[i]$. Also analogously, each set s_j can be accessed through $A_S[j]$. We consider a single slot of a Congest algorithm. A slot is divided into three parts. The parts are executed one after the other:

1. First, each node receives all messages from its neighboring nodes. This step is in the first slot omitted.
2. Second, they perform local computations on all received messages.
3. Third, they send new messages to its neighboring nodes.

These three steps can be emulated as follows. Assume that the simulation worked successfully until slot t. Hence, at the beginning of slot t, all sets and elements store all messages received until slot t in their local memory location. For the very first slot, this is trivially fulfilled because no message has been sent. Further, we iterate over all objects and perform the local computations that the set or element would execute in the Congest algorithm. As the object stored all messages sent in the previous slot, the results must be the same. More concrete, the simulation must generate all messages sent in slot t. For each message m_{ij} that u_i sends to its j^{th} set, we use the query `SetOf`(i, j) to obtain its index j' and then add the message to $A_S[j']$. The same is done vice versa for messages from sets to elements using `EltOf`(i, j). Hence, each message needs exactly one query. After all messages have been handled, A_E and A_S contain the nodes' states and all received messages. With this information, each node's action in the next slot of the distributed algorithm can be simulated. Hence, the sequential algorithm again iterates over A_E and A_S to compute the next slot's messages according to the algorithm. Thus, any distributed algorithm sending x messages can be transformed into a sequential algorithm with x queries. Hence, any lower bound on the queries in the sequential model is also a lower bound on the messages in the KT_0-Congest model. This only holds as long as nodes do not have identifiers that are known to their neighbors. Hence, if there were a distributed algorithm that solves SetCover with, say, $\tilde{o}(m\sqrt{\Delta})$ messages, it could be transformed in a sequential algorithm with $\tilde{o}(m\sqrt{\Delta})$ queries. Thus, any lower bound for the queries in the sequential model is a lower bound for the messages in our model. With this knowledge, we consider the lower bounds

from [20] and analyze the degree of their lower bound instances. Indyk et al. show that:

Lemma 10.7 (Lower bound from [20])**.** Let $\kappa \in \left[8, \tilde{O}\left(\left(\frac{n}{\log n}\right)^{\frac{1}{2\alpha}}\right)\right]$ be the size of an optimal solution for SetCover and let $\alpha \in [1, \log n]$ be an approximation factor. Then, for an optimal solution size κ and a approximation factor α there are families instances of $\mathscr{I}_1(\alpha, \kappa)$ and $\mathscr{I}_2(\alpha, \kappa)$, such that:

1. There is an instance $I \in \mathscr{I}_1(\alpha, \kappa)$, s.t., any randomized algorithm that yields an α-approximation with probability greater than $\frac{2}{3}$, it needs at least $\tilde{\Omega}\left(m\left(\frac{n}{\kappa}\right)\right)$ queries for $\alpha < 1.01$.
2. There is an instance $I \in \mathscr{I}_2(\alpha, \kappa)$, s.t., any randomized algorithm that yields an α-approximation with probability greater than $\frac{2}{3}$ needs at least $\tilde{\Omega}\left(m\left(\frac{n}{\kappa}\right)^{\frac{1}{2\alpha}}\right)$ queries for any $\alpha \geq 1.01$.

At the core of these lower bound constructions is a certain distribution of two problem instances, called *median instance* and *modified instance*. One argument from [20] is that an algorithm needs many queries to distinguish between a median instance and a modified instance. Both types of instances have a differently sized optimal solution. Thus, a good approximation algorithm enables us to distinguish them based on its output. More concrete, they show that a deterministic algorithm executed on an instance drawn from the distribution of median and modified instances, requires in expectation many queries to determine whether it was given a median or modified instance. Then, they use Yao's principle [42] to prove that for every randomized algorithm there is an instance from the support of these distributions. This instance requires at least that many queries. In the following, we abuse notation. We refer to these distributions as $\mathscr{I}_1(\alpha, \kappa)$ and $\mathscr{I}_2(\alpha, \kappa)$. We prove that, w.h.p., both distributions return an instance of degree $\tilde{\Theta}(n\kappa)$ and $\tilde{\Theta}((n\kappa)^{\frac{1}{2\alpha}})$ respectively. Thus, there are worst case instances that have these degrees. This follows because instances of higher degrees would *not* be drawn from the distribution w.h.p. Thus, they do not significantly affect the expected performance of the deterministic algorithm as they only make a small contribution to the expectation. We consider the distributions $\mathscr{I}_1(\alpha, \kappa)$ and $\mathscr{I}_2(\alpha, \kappa)$ conditioned on the event that we only draw instances of suitable degrees $\tilde{\Theta}(n\kappa)$ and $\tilde{\Theta}((n\kappa)^{\frac{1}{2\alpha}})$. Due to the law of total expectation, the expected performance of the deterministic algorithm would asymtotically be the same on the conditioned instances. We apply Yao's principle. Thus, there must be a worst-instance amoung the conditioned instances. Let us analyze the degrees of these instances that are not explicitly shown in [20].

Assume we want to build a distribution over instances with sets $\mathscr{S}$ and elements $\mathscr{U}$. The median instance of these sets and elements looks as follows: For each set $s \in \mathscr{S}$ and element $e \in \mathscr{U}$, independently add e to s with some probability $p \in [0, 1]$. Equivalently, we can consider the edges of the problem graph: Each possible edge (e, s) between an element and a set is independently added with probability p. This

construction works for any choice of $\mathscr{S}$ with $|\mathscr{S}| := m$ and $\mathscr{U}$ with $|\mathscr{U}| := n$ as we distribute a fixed number elements to a fixed number of sets. We assume $m = n$ as it will simplify some calculations. The lemmas can be extended to support more values of m and n. The choice of p depends on the desired size of the optimal solution and the approximation ratio α. More concrete, p is chosen such that the optimal solution of the median instance is bigger than $\alpha\kappa$ for some $\kappa > 1$. In addition, the choice of p will ensure that there are no empty sets and each element is contained in some set w.h.p.

We can construct the modified instance as follows: We construct a median instance. Then, we pick some sets $(s_1, \ldots, s_\kappa)$ that we want to turn into the optimal solution. The conversion can be done as follows: As long as these sets do not contain all elements, a random pair (e, e') is swapped between sets. More concrete, an element e that still needs to be covered by $s_1, \ldots, s_\kappa$ is removed from a set s that contains it and added to s_κ. Likewise, an element e' contained in s_κ, but already covered by any set $s_1, \ldots, s_{k-1}$, is added to s in return. e and e' are chosen uniformly at random from all applicable elements. This process does not change the sets' and elements' degrees. With the help of this construction, the size of the optimal solution tells us whether we have a median or modified instance. If the solution is much bigger than κ, i.e., bigger than $\alpha\kappa$, we have a median instance; otherwise, it we have a modified instance.

The median and modified instances are used differently in $\mathscr{I}_1(\alpha, \kappa)$ and $\mathscr{I}_2(\alpha, \kappa)$. We prove a general lemma on the degree of a median instance. After that, we go into the concrete constructions of the distributions The expected degree of the median instance is strongly tied to the choice of p. Using the both directions of the Chernoff bound, we prove the following statement:

Lemma 10.8. Assume that $m = n$. Let the elements of a median instance be sampled with probability $p \in [0, 1]$. Let Δ be the maximum degree of the instance's communication graph. Then, for $np \geq c \cdot 12 \cdot \log n$, it holds:

$$Pr[\Delta \in \Theta(np)] \geq 1 - n^{-c} \tag{10.33}$$

Determine the maximum degree of all sets. Fix a set $s \in S_i$ and let $X_{(e,s)}$ be the event that $e \in \mathscr{U}$ is added to s. Hence, the expected degree of s is

$$\Delta_s = \sum_{e \in \mathscr{U}} X_{(e,s)} = \sum_{e \in \mathscr{U}} X_{(e,s)} = \sum_{e \in \mathscr{U}} p = np. \tag{10.34}$$

As all variables $X_{(e,s)}$ are independent, we apply the Chernoff bound and obtain:

$$Pr[\Delta_s \leq \frac{1}{4}np] \leq Pr[\Delta_s \leq \frac{1}{4}\Delta_s] \tag{10.35}$$

$$\leq Pr[\Delta_s \leq \left(1 - \frac{3}{4}\right)\Delta_s] \tag{10.36}$$

$$\leq e^{-\frac{3^2 np}{4^2 \cdot 2}} = e^{-\frac{9np}{32}}. \tag{10.37}$$

For $np \geq c \cdot 12 \cdot \log n$, we have:

$$Pr[\Delta_s \leq \frac{1}{4}np] \leq e^{-\frac{9np}{32}} \leq e^{-\frac{9 \cdot 12 \cdot c \log n}{32}} \leq e^{-3c \log n} \tag{10.38}$$

Analogously, for the other direction, we have:

$$Pr[\Delta_s \geq 2np] \leq Pr[\Delta_s \geq 2 \cdot \Delta_s] \leq Pr[\Delta_s \geq (1+1) \cdot \Delta_s] \leq e^{-\frac{np}{3}}. \tag{10.39}$$

Thus, for $np \geq c \cdot 12 \cdot \log n$, we have:

$$Pr[\Delta_s \geq 2np] \leq e^{-3c \log n} \tag{10.40}$$

By taking the union bound over all sets, we can conclude that $\Delta \in \Theta(np)$ w.h.p. The same holds for the elements' degrees as we assumed $n = m$. We explain the concrete construction of the two lower bound instances promised by Lemma 10.7 and prove that their degrees are $\Theta(n\kappa)$ and $\Theta((n\kappa)^{\frac{1}{2\alpha}})$ respectively. The proofs will only hold for specfic small values of κ. More concrete, they hold for a lower range of values compared to [20]. They show the existence of hard instances with high degrees. We will describe the construction and *not* repeat the full lower bound argument because it is out of scope for this article. For more details, see [20]. Let us consider the construction of the first lower bound distribution. We have:

Lemma 10.9. Let $\mathscr{I}_1(\alpha, \kappa)$ be family/distribution of instances given in Lemma 10.7. For $\alpha < 1.01$ and $\kappa \in \left[8, \left(\frac{n}{24\alpha \log n}\right)\right]$ an instance of $\mathscr{I}_1(\alpha, \kappa)$ has degree $\Delta \in \Theta(n\kappa)$, w.h.p.

We can describe the construction of an instance $I \in \mathscr{I}_1(\alpha, \kappa)$. Note that our goal is to construct an instance with m sets and n elements. In every instance $I \in \mathscr{I}_1(\alpha, \kappa)$, the elements are divided into $t \in \Theta(\kappa)$ subinstances $(I_1, \ldots, I_t)$. We assume w.l.o.g. that t divides n and m. Thus, we only deal with integers. Each I_i is either a median instance or a modified instance with $m' := m/t$ sets and $n' := n/t$ elements. In each subinstance, each element adds itself to a set with probability $p = 1 - \sqrt{9 \log n' n'}$. Hence, all elements of this subinstance can be covered by 3 sets. Then, some of the subinstances get modified such that their optimal solution reduces to 2. The modification does not change the maximum degree of the corresponding communication graph, see above. We focus on the median instance.

There are no edges between the subinstances. Hence, we consider a fixed instance and compute its degree using Lemma 10.8. To do so, we prove that the expected degree is big enough to apply the Chernoff bound. We chose $p = 1 - \sqrt{9 \log n' n'}$. For a big enough n', it holds $p \geq 12$. We have $t \in \Theta(\kappa)$. An appropriate choice of κ therefore yields that $t \leq \frac{n}{c \cdot 24 \cdot \log n}$. Thus, it holds that:

$$n'p \geq \frac{n'}{2} = \frac{n}{2t} \geq \frac{c \cdot 24 \cdot \log n}{2} = c \cdot 12 \cdot \log n. \tag{10.41}$$

This expected value is clearly big enough for Lemma 10.8 to hold. Thus, w.h.p, the degree of each subinstance can be bounded as follows:

$$\Delta \in \Theta(n'p) \subseteq \Theta\left(n'\left(1-\sqrt{\frac{9\log n'}{n'}}\right)\right) \subseteq \Theta(n') \subseteq \Theta(nt) \subseteq \Theta(n\kappa) \tag{10.42}$$

Here, we used that $t \in \Theta(\kappa)$. Finally, we need to prove that every sequential algorithm needs $\Omega(n\Delta)$ queries. The lower bound from Lemma 10.7 implies that each algorithm needs $\Omega\left(n^2k\right)$ queries on this instance. As $\Delta \in \Omega(n\kappa)$, any algorithm must make $O\left(n\Delta\right)$ queries to obtain an α-approximation with probability $\geq \frac{2}{3}$. This proves the first lower bound and shows that in order to get arbitrarily close to the optimal solution, sets and elements need to communicate with almost all their neighbors (in certain instances).

We consider the other lower bound that handles less precise approximations. The next lower bound construction is summarized in the following lemma:

Lemma 10.10. Let $\mathscr{I}_2(\alpha,\kappa)$ be the family/distribution of instances given in Lemma 10.7. For $\alpha \geq 1.01$ and $\kappa \in \tilde{O}(1)$ an instance of $\mathscr{I}_2(\alpha,\kappa)$ has degree $\Delta \in \tilde{\Theta}(n\kappa)$, w.h.p.

The distribution for this lower bound consists of a single median instance that may or may not be modified. For the median instance, we add an element to set with probability

$$p := 1 - \left(\frac{8(\alpha\kappa+2)\log n}{n}\right)^{\frac{1}{4\alpha+1}}. \tag{10.43}$$

For the modification, we swap elements until we have an optimal solution of size κ. The modification does not change the degree. We only focus on the median instance. We prove that the (expected) degree is big enough to apply the Chernoff bound and then provide an upper bound.

Lower Bound We provide a lower bound for the expectation because it needs to be big enough, i.e., $\Omega(\log n)$ for Lemma 10.8 to hold. As $\kappa \leq \left(\frac{n}{16\alpha\log n}\right)^{\frac{1}{4\alpha+1}}$ and $\kappa > 2$, we have:

$$\kappa^{4\alpha} := \frac{\kappa^{4\alpha+1}}{\kappa} \leq \frac{1}{\kappa}\left(\frac{n}{16\alpha\log n}\right) \tag{10.44}$$

$$= \frac{n}{16\kappa\alpha\log n} \leq \frac{n}{8(\kappa+2)\alpha\log n} \tag{10.45}$$

Thus, the probability p can be approximated as follows:

$$p = 1 - \left(\frac{8(\alpha\kappa + 2)\log n}{n}\right)^{\frac{1}{\alpha k}} \tag{10.46}$$

$$\geq 1 - \left(\frac{1}{\kappa^{4\alpha}}\right)^{\frac{1}{\alpha\kappa}} \geq 1 - \kappa^{-\frac{4}{\kappa}} \tag{10.47}$$

$$= 1 - e^{\frac{4\log\kappa}{\kappa}} \tag{10.48}$$

By our assumption that $\kappa \geq 8$, it holds that $\frac{4\log\kappa}{\kappa} < 1.5$. For any $x < 1.5$, it holds that $e^{-x} < 1 - \frac{x}{2}$ and therefore, we have:

$$p \geq 1 - e^{\frac{4\log\kappa}{\kappa}} \geq 1 - \left(1 - \frac{2\log\kappa}{\kappa}\right) \tag{10.49}$$

$$= \frac{2\log\kappa}{\kappa} \tag{10.50}$$

Thus, for the expected number of elements in a set, it holds:

$$E[\Delta_s] = np \geq n\frac{2\log\kappa}{\kappa} \in \Omega\left(\frac{n}{\kappa}\right) \tag{10.51}$$

Further, as $\kappa \in O\left((n\log n)^{\frac{1}{5}}\right) \subseteq o(n\log n)$, we have $p \in \omega\left(\frac{\log n}{n}\right)$ and in particular $np \geq c \cdot 12 \cdot \log(n)$ for any c (provided that n is big enough).

Upper Bound For the upper bound of $E[\Delta_s]$ we conclude that we have $E[\Delta_s] = np \in O(n)$ as $p < 1$. If κ is constant, it holds $E[\Delta_s] \in O(n\kappa)$ as desired. Furthermore, for a polylogarithmic κ, it holds $E[\Delta_s] \in \tilde{O}(n\kappa)$.

Finally, we apply Lemma 10.8 because we have a lowerbound of $\Omega(n\kappa)$ and an upper bound of $\tilde{O}(n\kappa)$. The degree of this instance is $\Delta \in \tilde{\Theta}(n\kappa)$ w.h.p. Thus, we obtain the lemma. Hence, an instance is of degree $\Delta \in \tilde{\Theta}(n\kappa)$, w.h.p., and any algorithm must make $\tilde{\Omega}\left(n\Delta^{\frac{1}{2\alpha}}\right)$ queries to obtain an α-approximation with probability $\geq \frac{2}{3}$ by Lemma 10.7. This finishes our analysis.

11 Conclusion and Future Work

Conclusion
In this thesis, we introduced the hybrid communication model in order to find c-competitive routing paths in ad hoc networks. We assumed that only geographical coordinates of the source and the target are known. To avoid routing into radio holes, we considered convex hulls and bounding boxes as radio hole abstractions. We assumed that the convex hulls to not intersect. In addition, we considered two different types of intersections of bounding boxes, i.e., pairwise intersecting bounding boxes and multiple bounding boxes.

For convex hulls, we assumed that any distribution of the nodes in V ensure that UDG(V) is connected and of bounded degree. We designed an algorithm that computes an abstraction of UDG(V) in $\mathcal{O}(\log^2 n)$ communication rounds. This algorithm uses only polylogarithmic communication work at each node so that c-competitive paths between all source-destination pairs can be found in an online fashion.

Also for non-intersecting bounding boxes, we designed a routing strategy to route in the underlying ad hoc network. This strategy achieves provably better results than any previous online routing strategy for geometric ad hoc networks. Our strategy finds 18.55-competitive paths between source and destination pairs that are positioned outside of bounding boxes. In the case of intersecting bounding boxes, we considered the case of pairwise intersecting bounding boxes and presented and proved a routing strategy that finds 28.83-competitive paths between any pair of sources and destinations outside of bounding boxes. In addition, in the case of multiple intersecting bounding boxes, we provided a $(10.68 + c \cdot 12.83)$-competitive routing strategy that provides c-competitive paths between representatives of outer intersection points of bounding boxes.

Further, we presented message- and energy-efficient distributed algorithms for Set-Cover in the $KT0$ model. This KT_0 algorithm has polylogarithmic runtime and approximation ratio while sending only $\tilde{O}(\sqrt{\Delta}(n+m))$messages, w.h.p.. An event holds with high probability, if it holds with probability $1 - o(n^c)$ for some $c > 1$. In addition, we give arguments that one cannot hope for far better results as we show that there are instances that require $O(\sqrt{\Delta}n)$ messages for a constant approximation. Finally, we proved that any algorithm that gives an $\mathcal{O}(1)$-approximation for SetCover needs at least $\omega(m\sqrt{n})$ messages on certain graph classes in the $KT0$

model.

Future Work

Following research could investigate how to route on competitive paths through intersecting convex hulls, see the following figure.

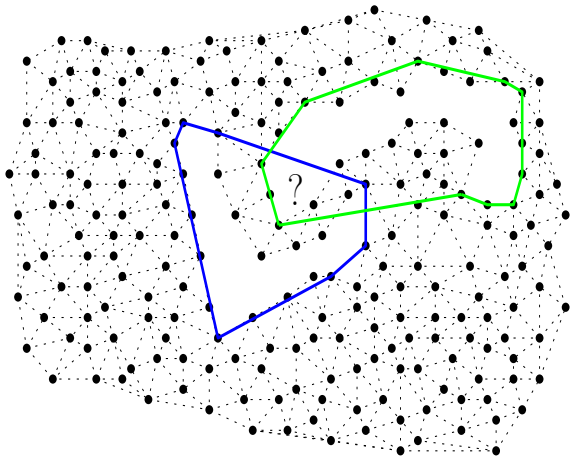

Figure 11.1: Intersecting convex hulls

To investigate this, one could try out some ideas of routing through intersecting bounding boxes.

One could define node movement as a time interval, where nodes move less than $\frac{1}{2}$. One can assume that only edges of length less or equal than $\frac{1}{2}$ are valid ad hoc links. This ensures that ad hoc links chosen by the routing protocol remain valid for the rest of the time interval, because all nodes which have been in the communication range of each other at the beginning of the time interval stay inside of these communication ranges, see the following figure.

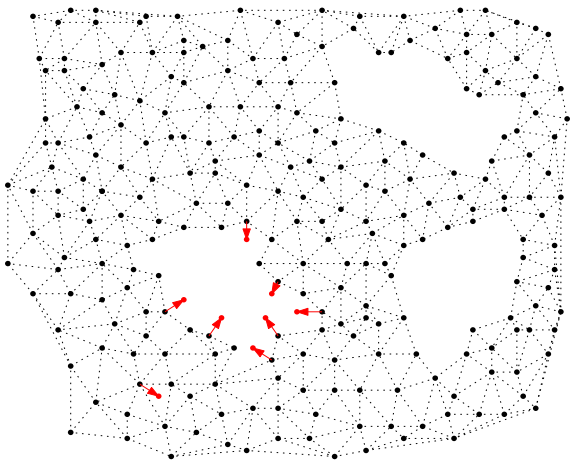

Figure 11.2: Hybrid communication model with node movement

In every time interval, a re-execution of all protocols except the protocol for the distributed overlay tree allows us to find c-competitive paths in a scenario where nodes change their positions. Further research could analyze how to maintain the overlay network efficiently if participants are able to change their positions or, additionally, one could consider joining and leaving nodes. Therefore, a model with bounded movement speed could be designed in which only parts of the overlay network have to be recomputed.

Further interesting research directions include energy efficient routing or load balancing to avoid congested convex hulls nodes and bounding box nodes. There is related work done on load balancing in restricted Delaunay graphs by Jie Gao, see [15]. There, the authors design an algorithm that calculates a paths that is close to the medial axis graph. However, this work is not directly applicable for our setting, where the goal is also to route on competitive path.

The application of this work is to route messages from a source to a target smartphone. Because security and safety play an increasingly important role in everydays live of smartphone users, it would be highly interesting to develop routing algorithms that besides calculating competitive paths are also provable secure. In terms of safety, there is related work on selfstabilzing Delaunay triangulations, see [21]. There the algorithm, once a Delaunay triangulation is calculated, the structure of this graph is preserved which is called a closure. But what about security in terms of anonymity? In the presented algorithm of this work, nodes know all intermediate routing targets between two convex hull nodes or bounding box nodes. But how can be ensured that nodes stay anonymous in front of other nodes such that their geometrical position and IDs are anonymously secure?

The routing approach in this thesis, is highly geometrical. A very different approach that is completely independent from geometry will be a new point of view for future work.

Also, our model does not tackle physical aspects of wireless communication. Interesting aspects are, for example, wireless interference in crowded areas.

For the $KT0$ model, a tighter lower bound would be interesting as the current bound only works for constant factor approximations, which are not achievable on many instances of the problem unless P=NP.

Lastly, it would be interesting to see whether the existence of unique identifiers known to all neighbors (i.e., the $KT1$-model) can improve the message complexity or if similar lower bounds hold.

List of Figures

Bibliography

[1] Nadeem Ahmed, Salil S. Kanhere, and Sanjay K. Jha. The holes problem in wireless sensor networks: a survey. *Mobile Computing and Communications Review*, 9(2):4–18, 2005.

[2] John Augustine, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, and Philipp Schneider. Shortest paths in a hybrid network model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1280–1299. SIAM, 2020.

[3] Nicolas Bonichon, Prosenjit Bose, Jean-Lou De Carufel, Vincent Despré, Darryl Hill, and Michiel H. M. Smid. Improved routing on the delaunay triangulation. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 22:1–22:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[4] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[5] Jannik Castenow, Christina Kolb, and Christian Scheideler. A bounding box overlay for competitive routing in hybrid communication networks. In Keren Censor-Hillel and Michele Flammini, editors, *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings*, volume 11639 of *Lecture Notes in Computer Science*, pages 345–348. Springer, 2019.

[6] Jannik Castenow, Christina Kolb, and Christian Scheideler. A bounding box overlay for competitive routing in hybrid communication networks. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, pages 14:1–14:10. ACM, 2020.

[7] Gianluca Cena, Adriano Valenzano, and Stefano Vitturi. Hybrid wired/wireless networks real-time industrial networks. In Richard Zurawski, editor, *Networked Embedded Systems - Volume 2 of the Embedded Systems Handbook*, page 26. CRC Press, 2009.

[8] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2010.

[9] Douglas Couto and Robert Morris. Location proxies and intermediate node forwarding for practical geographic forwarding. 09 2001.

[10] Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Improved leader election for self-organizing programmable matter. In Antonio Fernández Anta, Tomasz Jurdzinski, Miguel A. Mosteiro, and Yanyong Zhang, editors, *Algorithms for Sensor Systems - 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vienna, Austria, September 7-8, 2017, Revised Selected Papers*, volume 10718 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2017.

[11] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[12] Fabien Dufoulon, Janna Burman, and Joffroy Beauquier. Beeping a Deterministic Time-Optimal Leader Election. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[13] Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing holes in sensor networks. *MONET*, 11(2):187–200, 2006.

[14] Michael Feldmann, Kristian Hinnenthal, and Christian Scheideler. Fast hybrid network algorithms for shortest paths in sparse graphs, 07 2020.

[15] Jie Gao and Li Zhang. Trade-offs between stretch factor and load-balancing ratio in routing on growth-restricted graphs. *IEEE Trans. Parallel Distributed Syst.*, 20(2):171–179, 2009.

[16] Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, and Christian Sohler. Distributed monitoring of network properties: The power of hybrid networks. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 137:1–137:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[17] Robert Gmyr and Gopal Pandurangan. Time-message trade-offs in distributed algorithms. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA,*

October 15-19, 2018, volume 121 of *LIPIcs*, pages 32:1–32:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[18] Thorsten Götte, Christina Kolb, Christian Scheideler, and Julian Werthmann. Beep-and-sleep: Message and energy efficient set cover. In Leszek Gasieniec, Ralf Klasing, and Tomasz Radzik, editors, *Algorithms for Sensor Systems - 17th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2021, Lisbon, Portugal, September 9-10, 2021, Proceedings*, volume 12961 of *Lecture Notes in Computer Science*, pages 94–110. Springer, 2021.

[19] Thorsten Götte, Christina Kolb, Christian Scheideler, and Julian Werthmann. Beep-and-sleep: Message and energy efficient set cover. *CoRR*, abs/2107.14570, 2021.

[20] Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Set cover in sub-linear time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2467–2486. SIAM, 2018.

[21] Riko Jacob, Stephan Ritscher, Christian Scheideler, and Stefan Schmid. A self-stabilizing and local delaunay graph construction. pages 771–780, 12 2009.

[22] Lujun Jia, Rajmohan Rajaraman, and Torsten Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Comput.*, 15(4):193–205, 2002.

[23] Daniel Jung, Christina Kolb, Christian Scheideler, and Jannik Sundermeier. Brief announcement: Competitive routing in hybrid communication networks. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 231–233. ACM, 2018.

[24] Daniel Jung, Christina Kolb, Christian Scheideler, and Jannik Sundermeier. Competitive routing in hybrid communication networks. In Seth Gilbert, Danny Hughes, and Bhaskar Krishnamachari, editors, *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11410 of *Lecture Notes in Computer Science*, pages 15–31. Springer, 2018.

[25] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM 2000, Proceedings of the sixth annual international conference on Mobile computing and networking, Boston, MA, USA, August 6-11, 2000.*, pages 243–254, 2000.

[26] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry, UBC, Vancouver, British Columbia, Canada, August 15-18, 1999*, 1999.

[27] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 980–989. ACM Press, 2006.

[28] Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 25–32. ACM, 2003.

[29] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 63–72. ACM, 2003.

[30] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2002), Atlanta, Georgia, USA, September 28-28, 2002*, pages 24–33. ACM, 2002.

[31] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2003, Annapolis, Maryland, USA, June 1-3, 2003*, pages 267–278. ACM, 2003.

[32] Xiang-Yang Li, G. Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1268–1277 vol.3, 2002.

[33] Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 192–201. ACM, 2015.

[34] R. Miller and Q. F. Stout. Efficient parallel convex hull algorithms. pages 1605–1618, 12 1988.

[35] Damon Mosk-Aoyama, Tim Roughgarden, and Devavrat Shah. Fully distributed algorithms for convex optimization problems. *SIAM J. Optim.*, 20(6):3260–3279, 2010.

[36] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, USA, 2000.

[37] R. Tyrrell Rockafellar. 1970.

[38] Stefan Rührup and Christian Schindelhauer. Online multi-path routing in a maze. In Tetsuo Asano, editor, *Algorithms and Computation, 17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006, Proceedings*, volume 4288 of *Lecture Notes in Computer Science*, pages 650–659. Springer, 2006.

[39] Christian Scheideler, Andréa W. Richa, and Paolo Santi. An o(log n) dominating set protocol for wireless ad-hoc networks under the physical interference model. In *Proceedings of the 9th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2008, Hong Kong, China, May 26-30, 2008*, pages 91–100. ACM, 2008.

[40] Jannik Sundermeier. *Routing in Hybrid Communication Networks with Holes - Considering Bounding Boxes as Hole Abstractions*. Universität Paderborn, 2017.

[41] Ge Xia. The stretch factor of the delaunay triangulation is less than 1.998. *SIAM J. Comput.*, 42(4):1620–1659, 2013.

[42] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977.

[43] Jiguo Yu, Lili Jia, Dongxiao Yu, Guangshun Li, and Xiuzhen Cheng. Minimum connected dominating set construction in wireless networks under the beeping model. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 972–980. IEEE, 2015.