Towards a Theoretical Analysis of the Routing Architecture KIRA

Master's Thesis of

Wendy Yi

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	TT.-Prof. Dr. Thomas Bläsius
Second reviewer:	Prof. Dr. Dorothea Wagner
Advisor:	TT.-Prof. Dr. Thomas Bläsius

October 29, 2022 – March 29, 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text. I also declare that I have read and observed the *Satzung zur Sicherung guter wissenschaftlicher Praxis am Karlsruher Institut für Technologie.*

Karlsruhe, March 29, 2023

. . . . . .

(Wendy Yi)

Abstract

KIRA is a recently developed routing architecture for communication networks. In the routing tier, KIRA implements the routing protocol R^2/Kad, which uses a virtual overlay network to find paths between different nodes of the network. Nodes of the overlay network are addressed by randomly assigned IDs, and the existence of virtual links between nodes heavily depends on the chosen IDs. As a consequence, the overlay network is completely independent of the underlying topology, which seems to be very advantageous. Experiments on different types of networks suggest that KIRA is highly scalable and robust. Despite that each node only has a very local view of the network (and thus, requires little memory), KIRA finds short paths in most of the evaluated networks.

In this thesis, we analyze properties of KIRA from a graph-theoretical perspective, where we focus on connectivity and stretch of the paths found by KIRA. We show that KIRA implements several mechanisms that are crucial to establish connectivity in certain networks. However, there are topologies that KIRA cannot connect with constant probability. Further, we investigate how modifications to the algorithm may improve the ability of KIRA to establish connectivity. To obtain a lower bound for the stretch of found paths, we construct a graph with random ID assignment, where KIRA finds a path that has $\Theta(\log(n))$ stretch with at least constant probability.

Zusammenfassung

KIRA ist eine neu entwickelte Routingarchitektur auf Kommunikationsnetzwerken. In der Routingschicht verwendet KIRA das Routingprotokoll R^2/Kad, das mit Hilfe eines virtuellen Overlay-Netzwerks Pfade zwischen verschiedenen Knoten des Netzwerks findet. Knoten des Overlay-Netzes werden durch zufällig gewählte IDs adressiert, wobei die virtuellen Links stark von diesen IDs abhängen. Dadurch ist das Overlay-Netzwerk vollkommen unabhängig von der darunter liegenden Topologie, was viele Vorteile mitzubringen scheint. Experimente auf verschiedenen Netzwerktypen legen nahe, dass KIRA skalierbar und robust ist. Obwohl jeder Knoten nur eine sehr lokale Sicht auf das Netzwerk hat (wodurch KIRA nur wenig Speicher benötigt), findet KIRA kurze Pfade in den meisten evaluierten Netzwerken.

In dieser Arbeit analysieren wir Eigenschaften von KIRA aus einer graphtheoretischen Perspektive. Hierbei beschäftigen wir uns mit der Konnektivität und dem Stretch von Pfaden, die durch KIRA gefunden werden. Wir zeigen, dass KIRA einige Mechanismen implementiert, die für manche Netzwerke notwendig sind, um Konnektivität herzustellen. Allerdings gibt es Topologien, für die KIRA mit konstanter Wahrscheinlichkeit keine Konnektivität herstellen kann. Außerdem untersuchen wir, wie sich verschiedene Änderungen am Algorithmus auf die Konnektivität auswirken. Für eine untere Schranke zum Stretch der gefundenen Pfade konstruieren wir einen Graphen mit zufällig gewählten IDs, auf dem KIRA mit mindestens konstanter Wahrscheinlichkeit einen Pfad mit $\Theta(\log(n))$ Stretch findet.

Contents

1 Introduction

Communication networks today get increasingly larger and more dynamic. With this, the demand for faster and more robust routing protocols on networks grows. One common approach is to separate the view on the network from its underlying topology so that a change in the physical network does not affect the virtual network. This is the idea of many *Distributed Hash Tables* (DHT), which are mostly used to store resources in a decentralized manner, for example in a peer-to-peer network. To locate the node of the network that stores some specific resource, the idea is to repeatedly get "closer" to it in the virtual network. Distances in the virtual network are usually independent of the physical network and are defined differently for different implementations of DHTs. The most widely used DHT is Kademlia, which was introduced in 2002 by Maymounkov and Mazières [MM02]. Nodes are uniquely addressed by IDs, which are chosen randomly for each node. Two IDs are closer to each other if the bitwise XOR interpreted as an integer is smaller.

Although DHTs are not originally designed for routing in networks, many routing protocols employ ideas that are based on DHTs. KIRA (Kademlia-directed ID-based Routing Architecture) is a routing architecture for networks based on Kademlia that was developed in 2022 by Bless et al. [BZDH22]. It consists of a routing tier and a forwarding tier, where the routing tier is responsible for establishing connectivity in the control plane and for finding paths between nodes of the network. The forwarding tier forwards data packets according to the paths determined by the routing tier. In this thesis, we only focus on the routing tier, which implements the routing protocol R^2/Kad. This protocol is based on R/Kademlia, a recursive variant of Kademlia that was introduced by Heep [Hee10]. As for Kademlia, each node has a randomly chosen ID and a node is found by recursively routing closer to its ID. An advantage of KIRA is that it is zero-touch, which means that nodes do not have to be manually configured if they join the network. In particular, routing tables, in which each node stores its virtual neighbors, are initially empty and filled over time. As these routing tables are limited in size, nodes only have a local view of the network, which makes KIRA very space efficient. Further, KIRA seems to be highly scalable and robust against both node and link failures. Experiments suggest that KIRA can establish connectivity in many cases and is able to find reasonably short paths between different nodes of the network [BZDH22].

On the evaluated network types, KIRA performs well, but it is interesting to see if this is also transferred to other network types and maybe even arbitrary graphs. Especially if KIRA aims to be standardized at some point in the future, guarantees on the quality of the algorithm for arbitrary graphs would be very useful. Although Kademlia and other Kademlia-based routing protocols have been extensively studied using simulations, there are only few works that analyze them from a theoretical perspective. In this thesis, we aim to understand certain properties of KIRA better, mainly focusing on the two aspects connectivity and the length of found paths. Both aspects are essential for most routing protocols, especially being able to establish connectivity is a minimum requirement for any routing protocol. After introducing basic concepts used in this thesis and describing the functionality of KIRA in detail in Chapter 2, we study how and if KIRA can establish connectivity in Chapter 3. We show in Section 3.2 that there are graphs which are not connectable by KIRA. In Section 3.3, we analyze how different

mechanisms implemented in KIRA contribute to establishing connectivity in a network. Further, we discuss in Section 3.4 how some modified versions of KIRA can connect both arbitrary graphs in certain scenarios and paths as a restricted graph class.

Finding short paths is also an important quality of routing protocols, as shorter paths can reduce network traffic and delay, among other advantages. Chapter 4 deals with the length of paths between different nodes found by KIRA, where we provide bounds for both worst case ID assignments (Section 4.2) and random ID assignments (Section 4.3).

1.1 Related Work

The routing protocol R^2/Kad that is implemented in the routing tier of KIRA is based on Kademlia. Kademlia is a peer-to-peer *Distributed Hash Table* (DHT) and was introduced in 2002 by Maymounkov and Mazières [MM02]. DHTs are used to store data in a decentralized manner, for example to facilitate file sharing in peer-to-peer networks. Like usual hash tables, data objects can be retrieved and stored using their corresponding keys. However, in a distributed hash table, each node is responsible for storing different parts of the data, where the key of an object determines its storage location in the network. In other words, DHTs define a mapping of objects to nodes in the network. To retrieve data, the storage location must first be found. To locate nodes in the network, Kademlia sets up an *overlay network* on top of the actual network. In the overlay network, each node is uniquely identified by an ID that is chosen uniformly at random. Additionally, Kademlia defines an ID-based metric in the overlay graph which uses the XOR-operator. Formally, the distance between two IDs x and y is the integer interpretation of the bitwise XOR of the two IDs.

The routing algorithm of KIRA is based on the algorithm of Kademlia for locating IDs in the network. Each node of the network stores its neighbors of the overlay network in its *routing table*. The neighbors of a node are not distributed evenly across the ID space, instead, a node has more neighbors in closer distance ranges than in ranges that are further away in the ID space. If a node requests a path to another node with a certain ID, it iteratively forwards the request to nodes that are closer to the target in the ID space until the target is found, or no closer node is known. A more formal and detailed description of the algorithm as it is used for KIRA can be found in Section 2.3. Note that Kademlia simply assumes connectivity of the underlying network and is only responsible for finding a path in the overlay network. How the paths in the underlay graph between different nodes are found, is not specified in the protocol. As a consequence, it depends on a routing protocol that additionally provides connectivity in the underlay graph.

Kademlia is used in many networks, and it has been extensively analyzed using measurements and simulations [SR06 | OHKY10 | BS07]. For other implementations of DHTs such as Chord [Sto+03] and CAN [Rat+01], there are analytical approaches to study resilience and routing distances in networks [LCW05 | WXZ05]. However, there is only a limited number of works that study Kademlia from a theoretical perspective. In the original Kademlia paper, Maymounkov and Mazières [MM02] state that Kademlia is able to find a path between any pair of nodes in at most $\log(n)$ steps, where n is the number of vertices in the graph, but they do not provide a rigorous proof. In 2013, Cai and Devroye [CD13] analyzed the number of steps to locate a given ID by interpreting a network with its IDs as a random graph. They introduced two different models in their work, namely a deterministic ID model and a random ID model. In the deterministic ID model, the IDs are fixed and not randomly chosen. Note that even though the IDs are fixed, the number of needed steps is still a random variable since

they assumed that the neighbors of a vertex are randomly chosen. For this model, they proved that the worst expected number of steps to route from a certain node to another node is $c \cdot \ln(n)$ for a constant $c \leq 1/\ln(2)$, thereby decreasing the upper bound given in the original Kademlia paper. Further, they obtained similar results for a fixed requesting vertex. In the random model, the IDs of the nodes are assumed to be chosen uniformly at random. Similar to the deterministic model, they showed that $c \cdot \ln(n)$ steps are sufficient to locate a node in the network with high probability [CD15]. In both models, the constant c only depends on k, a parameter that is fixed for a Kademlia-system.

However, Cai and Devroye only considered worst cases for the expected number of needed steps to locate a node in the overlay network. This gap is targeted by Roos et al. [RSS13], who introduced a theoretical framework to determine the hop count distribution of a given Kademlia-system without the need of simulations. They modeled routing in a Kademlia-system with certain given parameters as a *Markov chain*, where a state is a vector that consists of the contacted vertices that are closest to the target. They proposed an algorithm that provides both lower and upper bounds on the hop count distribution of a given Kademlia-system. Further, they proved that computing such a hop count distribution is efficient in both space and time. Although they assumed a network in a steady state (with no failures) for their initial model, they additionally introduced an extended model that also considers non-responding nodes and incomplete routing tables. Based on the results from this model, Salah et al. [SRS14a | SRS14b] suggest a new neighbor selection strategy to diversify the IDs of the neighbors, aiming to improve lookup performance of Kademlia.

The previously mentioned works mostly consider the connectivity of the overlay graph as given and assume that each node in the network knows sufficiently many neighbors in the overlay network. However, failing nodes and links, as well as new nodes joining the network, may lead to a change in the connectivity of a Kademlia-system. Heck et al. [HKW17] analyze the connection resilience of the overlay in the Kademlia against attackers and failures. They define the *network connectivity* as the minimum number of node-disjoint paths between any pair of nodes in the *connectivity graph*. In the connectivity graph, a directed edge (u, v) exists if node u knows v as a neighbor in the overlay. For different simulated scenarios, they evaluated how the network connectivity of the overlay graph changes over time. Further, they state that a Kademlia-system with network connectivity κ is still connected (i.e., Kademlia is able to find a path between any two nodes) if at most $\kappa - 1$ nodes are compromised. However, it is possible that there are two nodes between which Kademlia cannot find a route, even though they are connected in the connectivity graph. Kong et al. [KBR08] take this problem into account and introduce the *reachable component method* as a framework to analyze performances of different DHTs. This is used to determine the expected number of nodes that are reachable from a specific source using Kademlia if nodes are compromised with a certain probability.

Apart from KIRA, there are several other routing protocols that are based on DHTs and which rely on a Kademlia-like overlay network such as UIP [For04], Virtual Ring Routing [Cae+06] and VIRO [JCZ11]. These routing protocols mainly differ in the topology of the overlay graph and how paths between consecutive overlay hops are found. For example, Virtual Ring Routing sets up an overlay graph where vertices are arranged in a cyclic fashion and connected by virtual links. Malkhi et al. [Mal+09] analyzed the expected path length on 2-dimensional grids and proved that it is at most $\log(n) \cdot \mathrm{diam}(G)$, where $\mathrm{diam}(G)$ denotes the diameter of a graph G. For d-dimensional grids, they further gave a lower bound for the expected path length. As for Kademlia, there are very few works that analyze the routing algorithms previously mentioned from a theoretical perspective.

2 Preliminaries

In this chapter, we introduce some basic definitions and notations for graph and probability theory that are used throughout this thesis. We mostly follow the notation for graph and probability theory by Diestel [Die12] and Mitzenmacher et al. [MU05], respectively. Further, we give a detailed description of KIRA as we use it in this thesis.

2.1 Graph Theory

A *graph* $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$. We use $V(G)$ and $E(G)$ to denote the vertex and edge set of a graph G, respectively. We differ between *directed* and *undirected* graphs. In a *directed* graph, an edge $(u, v) \in E(G)$ is a tuple and goes from u to v, whereas in an *undirected* graph, $E(G)$ is symmetric, i.e., $(u, v) \in E(G)$ if and only if $(v, u) \in E(G)$. For both kinds, we denote an edge from u to v by uv. Further, we assume that graphs are simple, i.e., there is at most one edge between any two distinct vertices, and the graph contains no self-loops. Two vertices u and v are *adjacent* if they are connected by an edge $uv \in E$. The *neighborhood* $N(v)$ of a vertex $v \in V(G)$ denotes the set of vertices that are adjacent to v, i.e., it is $N(v) = \{u \in V(G) \mid vu \in E(G)\}$.

A *path* is a sequence of vertices $P = (v_1, \ldots, v_p)$ with $p \in \mathbb{N}$ where consecutive vertices are adjacent to each other, i.e., $v_i v_{i+1} \in E(G)$ for all $i \in \{1, \ldots, p-1\}$. The vertices v_1 and v_p are called the *endpoints* of path P. The *length* of a path is the number of edges it consists of, which is equal to $p - 1$. If not stated otherwise, we assume that paths are *simple*, i.e., vertices in a path are pairwise disjoint. A vertex $u \in V(G)$ is *connected* to some vertex $v \in V(G)$ if there exists a path in G with endpoints u and v. If every pair of vertices $(u, v) \in V(G) \times V(G)$ is connected, we say that G is *connected*. A directed graph with this property is also called *strongly connected*. If vertex u is connected to vertex v, then the length of the shortest path from u to v is the *distance* $\operatorname{dist}(u, v)$ between u and v. The *diameter* of an undirected, connected graph is the greatest distance between any two vertices.

2.2 Probability Theory

A *sample space* Ω is the set of all possible outcomes, which we call *elementary events*. The *union bound* is often helpful to determine a lower bound for the probability of the union of multiple events. For a finite or countably number of events $E_1, E_2, \ldots$, it is

$$\Pr[E_1 \cup E_2 \cup \ldots] \leq \sum_{i=1} \Pr[E_i]. \tag{2.1}$$

For mutually disjoint events, equality holds in the equation above. Two events $A, B \in \Omega$ are *independent* of each other if and only if $\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]$. We denote the *conditional probability* that some event A occurs given that event B occurs by $\Pr[A \mid B]$. For any two events A and B with $\Pr[B] > 0$, it is $\Pr[A \mid B] \cdot \Pr[B] = \Pr[A \cap B]$. Let the events

$E_1, \ldots, E_n$ be a partition of the sample space Ω, i.e., $E_1 \cup \ldots \cup E_n = \Omega$ and $E_i \cap E_j = \emptyset$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. The law of total probability states that the probability for some event B can be rewritten as

$$\Pr[B] = \sum_{i=1}^{n} \Pr[B \cap E_i] = \sum_{i=1}^{n} \Pr[B \mid E_i] \cdot \Pr[E_i] \tag{2.2}$$

using the formula for conditional probability. A *random variable* is a function $X \colon \Omega \to \mathbb{R}$, where $\Omega \neq \emptyset$ is a sample space. The *expected value* of a discrete random variable $X \colon \Omega \to \mathbb{N}$ is defined as

$$\mathbb{E}[X] = \sum_{k \in \mathbb{N}} \Pr[X = k] \cdot k.$$

Expectations are linear, i.e., for random variables $X_1, \ldots, X_n$ it is

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i],$$

regardless of whether the random variables are independent or not. Similarly to conditional probability, we define *conditional expectation*. For two random variables $Y \colon \Omega \to \mathbb{N}$ and $Z \colon \Omega \to \mathbb{N}$, it is

$$\mathbb{E}[Y \mid Z = z] = \sum_{y \in \mathbb{N}} y \cdot \Pr[Y = y \mid Z = z].$$

We use $\mathbb{E}[Y \mid Z]$ to denote a random variable that depends on Z with value $\mathbb{E}[Y \mid Z = z]$ for $Z = z$. It is often useful to bound the probability that the value of a random variable is far from its expected value. The following theorem is a much used variation of the original Chernoff bounds [Che52] that gives a bound for the multiplicative lower and upper tail.

Theorem 2.1 (Chernoff Bounds)**:** *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0, 1\}$. Let $X \coloneqq \sum_{i=1}^{n} X_i$ be the sum of the random variables and $\mu \coloneqq \mathbb{E}[X]$ the expected value of X. Then, for any $\alpha \geq 0$, we have*

$$\Pr[X \leq (1 - \alpha) \cdot \mu] \leq e^{\frac{-\alpha^2 \cdot \mu}{2}} \tag{2.3}$$

and

$$\Pr[X \geq (1 + \alpha) \cdot \mu] \leq e^{\frac{-\alpha^2 \cdot \mu}{2+\alpha}}. \tag{2.4}$$

2.3 Introduction to KIRA

In this section, we introduce KIRA [BZDH22] and its variants as we use it in this thesis. Note that in this thesis, $\log(x)$ denotes the logarithm to the base of 2, whereas $\ln(x)$ denotes the natural logarithm of x. KIRA has two different views on the same network: The *underlay graph* represents the actual network, where edges model physical links between two nodes. The *overlay graph* consists of the same vertices as the underlay graph, but edges represent logical connections. An edge between two vertices u and v in the overlay graph corresponds to a simple, unique path in the underlay graph between the same vertices, which is stored along with the edge. In this thesis, the underlay graph is always an undirected graph, whereas the overlay graph is always directed. The overlay graph is initially empty, and vertices may add edges by discovering paths to other vertices in the underlay graph. Newly discovered

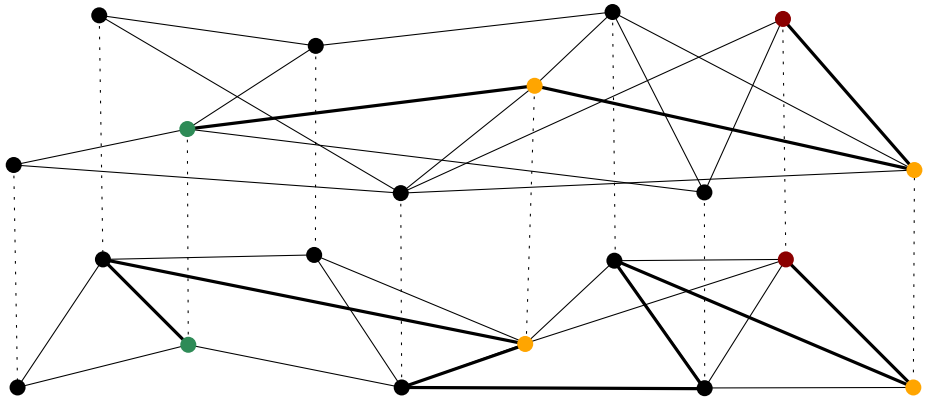

Figure 2.1: The underlay graph (in the lower half) represents the actual network, while the overlay graph (in the upper half) is a virtualized view on the graph constructed by KIRA. In this example, a path is found in the overlay graph from the green to the red vertex that goes over the overlay hops in orange. Each edge of the path corresponds to a unique path in the underlay graph.

vertices are then stored in the *routing table* of each vertex. The main idea of KIRA is to find a path in the overlay graph first, and then, to obtain the corresponding path in the underlay graph. An example can be seen in Figure 2.1, where a path is found in the overlay graph between the red and the green vertex, which goes over two *overlay hops*. In the underlay graph, each overlay edge of the found path is unpacked to the corresponding underlay path.

ID Space In the underlay graph, KIRA uses the usual distance metric on graphs, i.e., two vertices are closer in the underlay graph if the shortest path between them is shorter. In the overlay graph, IDs are used to define distances. Let $B \in \mathbb{N}$ be the number of bits in each ID. Then, the *ID space* $\{0,1\}^B$ consists of $N := 2^B$ distinct IDs. Initially, each vertex chooses its ID from the ID space independently and uniformly at random. We choose the size of the ID space large enough such that it is reasonable to assume that each vertex can be uniquely identified by its ID. This is closely related to the *birthday problem* which asks how likely it is that from a set of n randomly chosen people, two share the same birthday.

Lemma 2.2: *[STKT06] Suppose that there are q balls and m buckets, and a bucket is chosen uniformly at random for each ball. The probability that each bucket only contains one ball is at least*

$$1 - \frac{q \cdot (q-1)}{2m}.$$

With $q := n$ and $m := N$, where n is the number of vertices, this lemma implies that if we choose $N \in \omega(n^2)$, then IDs are unique with high probability. In this thesis, we assume that the ID space is sufficiently large, for example $N := n^3$.

The function $\mathrm{ID}\colon V(G) \to \{0,1\}^B$ returns the ID of each vertex. As for Kademlia, two IDs x and y are closer to each other in the ID space if they share a longer prefix. Formally, we define the *prefix bit distance* $\mathrm{dist}_p(x,y) := B - \mathrm{lcp}(x,y)$, where $\mathrm{lcp}(x,y)$ denotes the length of the *longest common prefix* of x and y. For the prefix bit distance of two vertices $u, v \in V(G)$, we use the same notation $\mathrm{dist}_p(u,v) := \mathrm{dist}_p(\mathrm{ID}(u), \mathrm{ID}(v))$. As this operation is symmetric, positive definite and satisfies the triangular inequality, it indeed defines a metric on the ID space. Note that the XOR metric used by Kademlia is finer than the prefix bit distance.

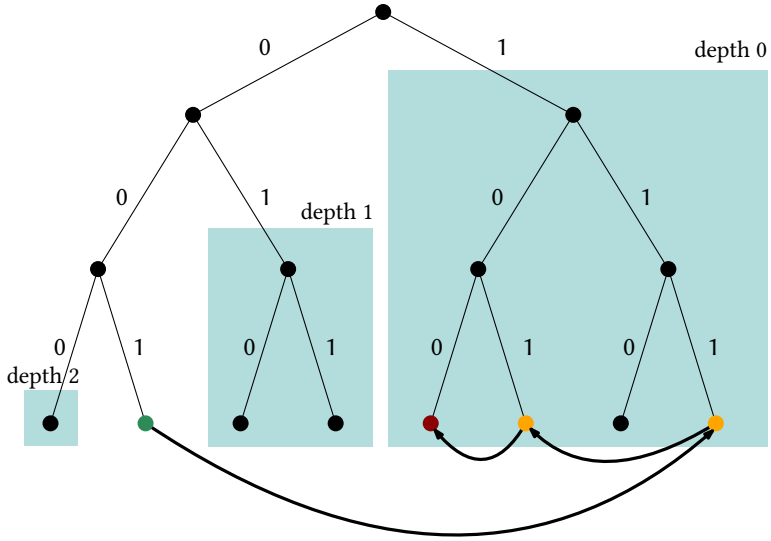

Figure 2.2: A trie with $B = 3$, where each vertex in the network corresponds to a leaf in the trie. Note that since not all IDs of the ID space are assigned, not every leaf has a corresponding vertex. The boxes represent the buckets of different depths in which contacts of the green vertex are stored. The overlay graph consists of directed edges between leaves. In this example, KIRA routes from the vertex with ID 001 to the vertex with ID 100 via the IDs 111, 101, and 100 by correcting bits one by one, starting with the most significant bit.

The overlay neighbors of a vertex v are stored in its routing table, where they are sorted by their prefix bit distance to v into B *buckets* of equal size. Roughly speaking, the existence of an edge between two vertices u and v in the overlay graph depends on the prefix bit distance of the respective IDs. More specifically, such an edge exists with higher probability if the IDs are closer to each other in the ID space, i.e., each vertex is supposed to have more neighbors in the overlay graph that are close in the ID space. Note that this is independent of the underlying topology and only depends on the chosen IDs.

We obtain another view on the ID space by interpreting IDs as leaves of a *binary trie* with depth B. Tries are data structures that are usually used to store and process strings. A binary trie is a rooted, balanced tree, where each inner vertex has exactly two children, which are labeled with 0 and 1, respectively. By following a path from the root to a leaf, we obtain the ID that corresponds to the leaf. The prefix bit distance of two leaves is $B - d$, where d is the depth of the *lowest common ancestor*. The lowest common ancestor is defined to be the root of the smallest subtrie that contains both leaves. Each vertex in the original graph corresponds to a leaf, and edges in the overlay graph connect leaves of the trie.

The idea of the routing algorithm is to find a requested ID in the overlay graph by greedily decreasing the prefix bit distance to the target at each overlay hop. In the trie, this means that a request is forwarded to a leaf whose lowest common ancestor with the requested ID is deeper than for the current leaf. At each overlay hop, the request gets closer to the requested ID, in the sense that the size of the smallest subtrie that contains both the current overlay hop and the requested ID decreases. A sketch of a trie with $B = 3$ can be seen in Figure 2.2.

It is possible that in some situations, there are multiple candidates for the next overlay hop. For instance, this may happen if several vertices would improve the prefix bit distance by the same number of bits. To uniquely choose the next overlay hop, a tie-breaking metric is

used. The *bit distance* between two IDs x and y is defined as $\text{dist}_b(x, y) = x \oplus y$, where the operator $\oplus$ denotes the bitwise exclusive or (XOR). Note that $\text{dist}_b(w, x) > \text{dist}_b(y, z)$ holds for IDs w, x, y and z if $\text{dist}_p(w, x) > \text{dist}_p(y, z)$. Further, for an ID x and each bit distance $d \in \{0, \ldots, 2^B\}$, there is exactly one ID y with $\text{dist}_b(x, y) = d$.

Routing Table The neighbors in the overlay graph of a vertex are called *contacts*, which are managed by each vertex in its routing table. Additionally, a corresponding *underlay path* is stored for each contact such that each vertex knows how to reach its contacts in the underlay graph. Contacts of a vertex are sorted by their prefix bit distance to the vertex in buckets of different *depths*. Each routing table consists of B buckets, where contacts in the bucket of depth $i \in \{0, \ldots, B-1\}$ of some vertex v have prefix bit distance $B - i$ to v. Thus, for every vertex v and every ID x, there is exactly one bucket in the routing table of v in which a vertex with ID x would be stored in. We denote this bucket by $B_v(x)$. Each bucket can only store a fixed number k of contacts, which we assume to be the same for all vertices and buckets. We further assume that k is independent of both the size of the ID space and the size of the graph if not stated otherwise. However, at some points, we use $k \in \Theta(\log(n))$ instead of a constant. This increases the space needed to store a routing table by a factor of $\Theta(\log(n))$. In practice, the number of contacts that can be stored in a bucket may vary for each vertex independently to improve performance. In the implementation of KIRA, 20 is usually used as a default value for k. Initially, the routing tables are empty, and they are filled with contacts over time. In the beginning, each vertex discovers its *3-hop-vicinity*, which consists of the vertices that have distance at most 3 in the underlay graph. All vertices in the 3-hop-vicinity of a vertex are then added to its routing table, with corresponding underlay paths. Routing table entries are continuously added, updated and replaced using certain mechanisms and strategies, which are explained in the Subsections 2.3.1, 2.3.2, and 2.3.3.

Path Request Like Kademlia, KIRA uses greedy routing by recursively forwarding a request to a contact that improves the bit distance to the requested ID in the ID space. The high level idea is to route to the requested ID by "correcting" bits in the ID of the requesting vertex one by one, starting with the most significant bit, until the requested ID is reached. Assume that some vertex v requests ID x. Vertex v sends the request to a contact in its routing table whose ID is bitwise closer to x than its own ID. It first tries to improve the prefix bit distance to the requested ID if possible. The contacts that are prefix-wise closer to x than v are exactly the contacts that are stored in the bucket $B_v(x)$ since this is the bucket that stores all contacts of v which share a longer prefix with x than v. If this bucket is not empty, v chooses one contact using a pre-defined *routing strategy* as the next *overlay hop*. If the bucket is empty, the prefix bit distance to the requested ID cannot be improved. In this case, v uses the XOR metric as a tie-breaker and selects the contact that is bitwise closest to the requested ID as the next overlay hop. The current overlay hop v then forwards the request to the chosen contact via the corresponding underlay path that is stored in the routing table entry. The next overlay hop then repeats this procedure and again sends the request to a contact that is bitwise closer to the requested ID. The taken underlay path of the request so far is forwarded along with the request and appended at each overlay hop. An example of the path of such a request can be seen in Figure 2.2, where the algorithm routes from ID 011 over the IDs 111 and 101 to the ID 100, correcting bits one by one.

Termination of Request If an overlay hop does not know a contact that is bitwise closer to the requested ID, the request terminates. This is naturally the case if the current overlay hop is assigned the requested ID, but it may also happen if the requested ID is not assigned to any vertex in the graph or if the algorithm could not find the requested ID. The final overlay hop then sends information back to the source of the request using the reversed underlay path. The information at least contains the path from the source to the final underlay hop.

2.3.1 Routing Strategies

A routing strategy is needed to uniquely select the next overlay hop if there are several candidates, which is the case if there are multiple contacts that improve the prefix bit distance. If there are no contacts that improve the prefix distance, the XOR metric is used to select the contact that is bitwise closest to the requested ID. In this thesis, we assume Proximity Routing is used if not stated otherwise. This is also the routing strategy used by the original algorithm.

Proximity Routing This strategy always chooses a contact with the shortest corresponding underlay path among the candidates which improve the prefix bit distance. If there are again multiple possible contacts, the XOR metric is used to uniquely select the candidate that is bitwise closest to the requested ID. This strategy is motivated by the desire to keep the underlay path to the target as short as possible by choosing a contact as the next overlay hop that is "close by" in the underlay graph.

ID Routing This strategy is not used by KIRA but is the usual routing strategy in Kademlia. It always chooses the (unique) contact that is bitwise closest to the target ID. This attempts to minimize the number of overlay hops on the path to the target since it tries to improve prefix-wise as much as possible. Further, the chosen contact from a bucket depends on the requested ID, which is not the case when using Proximity Routing. However, other than with Proximity Routing, the underlay paths between consecutive overlay hops might be longer.

2.3.2 Finding New Contacts

To establish connectivity, it is important to fill and improve routing tables. KIRA implements different mechanisms to find new vertices. Self-requests and probing describe different variants of requests, whereas closest-k-responses and path overhearing describe how vertices can learn new vertices using responses to requests. In the original variant of KIRA, self-requests, random probing, closest-k-responses and path overhearing is used.

Self-Requests The goal of each vertex is to fill its routing table as much as possible. Since fewer vertices belong to a deeper bucket of a vertex v, it is less likely that such a vertex is discovered by v and added as a contact. As a consequence, it seems that deeper buckets are harder to fill. To counter this issue, KIRA implements self-requests, where vertices repeatedly request their own IDs to find their closest neighbors in the ID space. Since the source equals the target in a self-request, the source itself is ignored at each overlay hop. The first overlay hop of a self-request is the bitwise closest contact of the requesting vertex. Necessarily, it is always bitwise further away from the target than the source. However, since after the first overlay hop, the bit distance to the requested ID strictly decreases at each overlay hop, a self-request always terminates.

Random Probing The goal of random probing is to discover new areas of the graph. In addition to vertices requesting their own IDs, vertices are also able to request random IDs. For random probing, each vertex requests randomly chosen IDs to fill its buckets and to test for connectivity in a later stage. Each ID of the ID space is chosen with the same probability. Vertices handle such a random probe exactly like a usual path request. In the original algorithm, each vertex repeatedly sends random probes with a fixed frequency.

Deterministic Probing Instead of probing randomly, vertices may also probe certain IDs deterministically by requesting a specific ID. For deterministic probing, it is important to be aware of the degree of knowledge vertices have. Originally, vertices have a local view of the network and therefore can only choose specific IDs based on their routing tables for probing. For instance, vertices can probe IDs that belong to buckets that are empty at the time of the probe without having to know more than their own routing table. However, at some points we assume that vertices know more about the graph than the information stored in their routing tables, and that they can request specific IDs arbitrarily. Since this is not realistic, it only serves a theoretical purpose, but is not useful in practice.

Via-Probing Via-probing is an enhancement of random and deterministic probing. With via-probing, a vertex that requests an ID can arbitrarily choose a contact to be the first overlay hop. Similar to self-requests, the first overlay hop might be bitwise further away from the requested ID than the source of the request, and the source is ignored at each overlay hop.

Closest-k-Response In DHTs, objects are stored at nodes whose IDs are closest to the object ID in the ID space. Thus, if an object ID is requested, the nodes with the closest IDs need to be found. KIRA, as it is implemented, differs between *exact* and *inexact* requests. Suppose that a request for ID x terminates at some vertex v. For an exact request, the response contains just the vertex v if it has ID x, otherwise an error is sent back to the source of the request since the requested ID was not found. For an inexact request, v sends back k contacts from its routing table that are bitwise closest to the requested ID x. In this thesis, we assume that all requests are inexact and that the final overlay hop of some request always responds with k contacts that are bitwise closest to the requested ID. If the final overlay hop does not know k contacts, it responds with all its contacts.

Path Overhearing With path overhearing, vertices can use any information which they forward in the underlay graph to update their routing tables. This information includes all vertices in the underlay path of the request and possibly more vertices from the response if closest-k-response is used. Path overhearing may be used to learn new contacts as well as to improve existing entries in the routing table.

2.3.3 Storing New Contacts

Suppose that a vertex v learns of a new vertex u. If the bucket $B_v(u)$ is not full yet and if it does not already contain u, then u is simply added to the bucket along with a corresponding underlay path from v to u. However, if $B_v(u)$ is already full, then v may replace another contact in $B_v(u)$ with u according to some *replacement strategy*. If no replacement strategy is

used, then u is discarded. The default replacement strategy used in this thesis is Proximity Neighbor Selection, which is also the strategy used by the original algorithm. Further, newly learned information can be used to improve existing entries in routing tables, for instance, if a shortcut to a contact is learned.

Proximity Neighbor Selection This strategy keeps the contacts with the shorter corresponding underlay path. Like the Proximity Routing strategy, this aims to minimize the underlay path lengths between two overlay hops. For tie-breaking, the XOR metric is used.

Propagation Propagation can be used instead of a replacement strategy and ensures that newly learned vertices can still be reached even if they are not stored in the own routing table. Suppose a vertex v learns of a new vertex u, but $B_v(u)$ is already full. Further, let $w \in B_v(u)$ be the contact to which v would forward a request with target u. Note that w depends on the routing strategy that is used. Then, v propagates u and a corresponding underlay path to u to w, which is now responsible for storing u as a contact. The underlay path to u is appended appropriately after every propagation. This step is repeated recursively until u is propagated to a vertex z that already knows u or where $B_z(u)$ is not full. In the latter case, the vertex z stores u as a contact with a corresponding underlay path. Note that this always terminates.

3 KIRA-Connectivity

In this chapter, we analyze the ability of KIRA to establish connectivity in the constructed overlay graph and to find a path between every pair of vertices. We first introduce the concept of KIRA-connectivity and study general properties of it. Then, we analyze KIRA with the originally implemented mechanisms. We show that there are graphs with constant diameter that are not KIRA-connectable with constant probability. For larger graphs, we prove that the connectivity mechanisms random probing and path overhearing are needed for certain instances to be able to establish connectivity. Further, we show that by slightly modifying KIRA, it is able to establish connectivity in specific scenarios, for instance, if a new vertex joins a KIRA-connected network. We end this chapter with studying paths specifically, for which we propose a strategy to establish KIRA-connectivity with high probability.

As we are often only interested in the most significant bits in the ID of a vertex, we simplify the notation by omitting irrelevant suffixes. More specifically, we say that some vertex v is a p-vertex for some bit string p if p is a prefix of $\mathrm{ID}(v)$. Further, we use the canonical notation for the concatenation of two bit strings. For example, for $\mathsf{p} = 10$, a $\mathsf{p}1$-vertex is a vertex whose ID starts with the prefix 101.

3.1 Properties of KIRA-Connectivity

For KIRA to find a path between all pairs of vertices, it is not sufficient to have a strongly connected overlay graph in the graph-theoretical sense since KIRA is only able to find paths in the overlay graph where the bit distance to the target is decreased at each overlay hop. Thus, we need a stronger concept of connectivity. Let G be a graph and $\mathcal{G}$ the state of the overlay graph with an ID distribution. We say that a vertex $u \in V(G)$ is *KIRA-connected* to a vertex $v \in V(G)$ if KIRA finds an overlay path to v when u requests $\mathrm{ID}(v)$. Further, the graph G is KIRA-connected if every pair of vertices is KIRA-connected to each other, i.e., if KIRA finds an overlay path between every pair of vertices. In this case, KIRA is always able to find the vertex that is bitwise closest to some requested ID.

KIRA-connectivity behaves quite differently compared to the usual connectivity of graphs. For instance, it is neither *symmetric* nor *transitive*. More formally, a relation $\mathcal{R} \subseteq V(G) \times V(G)$ is symmetric if $(u, v) \in \mathcal{R}$ implies $(v, u) \in \mathcal{R}$. It is transitive if for $(u, v) \in \mathcal{R}$ and $(v, w) \in \mathcal{R}$, (u, w) is in $\mathcal{R}$ as well. Moreover, adding new edges to the overlay graph (i.e., a vertex learns a new contact) can destroy the KIRA-connectivity for other vertices which were KIRA-connected before. On a similar note, deleting edges in the overlay graph (if they are not replaced) may establish KIRA-connectivity between some previously disconnected vertices. Edges are deleted if some links in the network fail, or if the corresponding contact in the overlay graph is non-responsive. The following lemma summarizes these properties.

Lemma 3.1: *KIRA-connectivity is not symmetric and not transitive in general. Further, KIRA-connectivity is not monotone under edge deletion.*

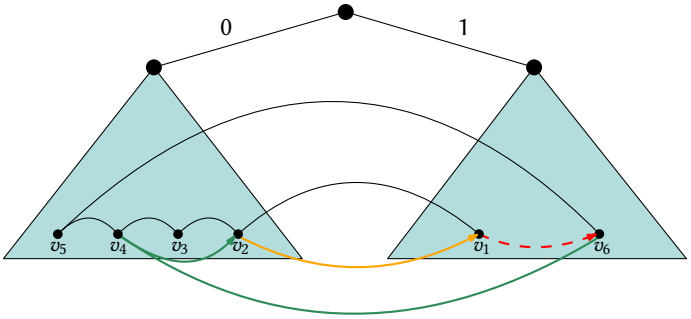

Figure 3.1: An example of an overlay graph, where KIRA-connectivity is neither symmetric nor transitive. Leaves of the trie are vertices, and the black edges are edges in the underlay graph. For the overlay graph, we assume that two vertices are adjacent if they have distance at most 3. A request from v_6 to v_2 may follow the green path, while a request from v_2 to v_1 or v_6 follows the orange path. However, v_1 is unable to find v_6.

Proof. We prove each part of the lemma by giving an example of an overlay graph where KIRA-connectivity is neither symmetric nor transitive. Let G be a path that consists of six vertices $v_1, \dots, v_6$. Suppose that v_1 and v_6 are 1-vertices and that $v_2, \dots, v_5$ are 0-vertices. Consider the state of the overlay graph immediately after the initial discovery of the 3-hop-vicinity, i.e., each vertex knows exactly the vertices with at most distance 3 as contacts. This situation is depicted in Figure 3.1. If v_6 requests $\mathrm{ID}(v_2)$, then this request reaches v_2 after one overlay hop in between, thus, v_6 is KIRA-connected to v_2. However, if v_2 requests $\mathrm{ID}(v_6)$, this request is sent to v_1, the only vertex in $B_{v_2}(v_6)$. But since v_1 does not know a vertex which is bitwise closer to v_6 than itself, the request terminates, and v_2 is not connected to v_6. Thus, KIRA-connectivity is not symmetric.

Further, v_6 and v_1 are not KIRA-connected, even though v_6 is KIRA-connected to v_2 and v_2 is KIRA-connected to v_1. Thus, KIRA-connectivity is not transitive for these vertices. Note that this example is independent of the chosen routing strategy.

Now, further assume that v_5 is bitwise (but not prefix-wise) closer to v_6 than v_2. Then, by deleting the overlay edge between v_1 and v_2, vertex v_2 is now able to find a path to v_6 via the overlay hop v_5, although it was previously not KIRA-connected to v_6. ∎

Despite these counter-intuitive properties of KIRA-connectivity, the following lemma shows that global KIRA-connectivity is implied by another property that only depends on the number of vertices in each bucket of each vertex. Suppose that every vertex v of a graph G has the following property: For every $i \in \{1, \dots, \log(N)\}$, vertex v knows a contact with prefix bit distance i if there is a vertex in the corresponding bit distance range, i.e., $B_v(u)$ is not empty for every $u \in V(G)$. We call an overlay graph with this property *well-filled*.

Lemma 3.2: *Every well-filled graph G is KIRA-connected.*

Proof. We assume that G is well filled, i.e., every vertex knows a contact with prefix bit distance i for each $i \in \{1, \dots, \log(N)\}$ if there is such a vertex in the corresponding bit distance range. Suppose that we want to route from some vertex v to some vertex t. Then, $B_v(t)$ is not empty by assumption, and $B_v(t)$ contains the contacts that are prefix-wise closer to t than v. If t is in $B_v(t)$, then KIRA directly routes to t, and the target is found. Otherwise,

KIRA chooses another contact $w \in B_v(t)$ and forwards the request to w, which improves the prefix bit distance to t by at least 1. By applying this argument inductively, we show that a path from v to t is found. ∎

Note that the previous lemma holds, regardless of the applied replacement and routing strategy. Since for a well-filled overlay graph, the prefix bit distance to the requested ID decreases by at least 1 at each overlay hop, the following holds.

Corollary 3.3: *If a graph G is well filled, then KIRA finds a path between two vertices u and v in at most* $\text{dist}_p(u, v)$ *overlay hops for any $u, v \in V(G)$.*

The reverse implication of Lemma 3.2 is not true in general since the prefix does not have to be improved at every overlay hop. It is possible that the bucket $B_v(t)$ at some overlay hop v is empty, but that the request can be forwarded to a contact that is still bitwise closer to t. As a consequence, not every bucket needs to contain a contact to ensure KIRA-connectivity. Note that in this case, if v is KIRA-connected to t, then the empty bucket $B_v(t)$ can easily be filled with t as a contact if v requests the ID of t. However, the following lemma states a condition that has to be satisfied if the graph is KIRA-connected.

Lemma 3.4: *Let G be a KIRA-connected graph. Then the following holds for all vertices u and v: If v is bitwise closer to u than every other vertex in G, then v must be a contact of u.*

Proof. Assume there are vertices u and v, where v is the bitwise closest vertex to u and where u does not know v as a contact. Then, KIRA cannot find a path from u to v since it only routes to contacts that are bitwise closer to the target than the source. ∎

3.2 Unconnectable Graph with Constant Diameter

In this section, we assume that KIRA uses the mechanisms self-request, path overhearing, deterministic probing, and k-closest-response. Apart from using deterministic instead of random probing, this is the original implementation of KIRA. In fact, we can even assume a stronger variant of deterministic probing, where each vertex has global knowledge of the graph. The results of this section are independent of the used replacement and routing strategy. In the following, we show that there exists a graph G with constant diameter which is assigned an ID distribution with constant probability such that KIRA cannot establish KIRA-connectivity for G. The graph G consists of two stars that are connected by a path. It has two vertex sets V_1 and V_2 that are separated by a path P of constant size $\ell \geq 4$ with vertices $p_1, \dots p_\ell$. The vertex sets V_1 and V_2 are independent sets, i.e., the subgraph induced by V_i contains no edge for $i \in \{1, 2\}$. Further, the vertices in the vertex sets V_1 and V_2 are adjacent to the endpoint p_1 and the endpoint p_ℓ of the path, respectively, forming one star at each endpoint. An example for such a graph can be seen in Figure 3.2. We show that if the IDs of G satisfy certain conditions, then G is not KIRA-connectable.

Lemma 3.5: *If the path P in graph G only consists of 0-vertices and if both V_1 and V_2 contain at least k 0-vertices and two 1-vertices each, then KIRA cannot establish connectivity for G.*

Proof. We prove that a 1-vertex in V_1 never learns of a 1-vertex in V_2 and vice versa by showing that this invariant holds independently of the issued requests. With $\ell \geq 4$, this invariant trivially holds after the discovery of the 3-hop vicinity. Consider some vertex $u \in V(G)$ that requests an ID x. We assume that the proposed invariant holds before the request and show

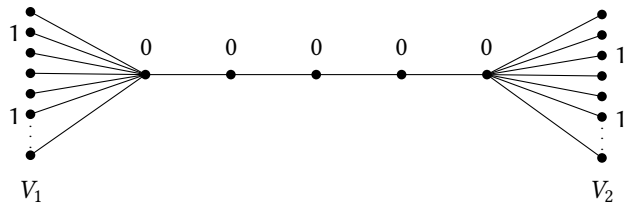

Figure 3.2: A graph with an ID assignment that is not KIRA-connectable. The parts on both ends of the path form a star. Each star contains at least two 1-vertices and k 0-vertices. The bits indicate if a vertex is a 1-vertex or a 0-vertex. 1-vertices of the left star cannot learn of 1-vertices of the right star.

that it still holds after the request. As KIRA only finds simple paths, such a found path never contains a vertex in V_1 or V_2 that is not the endpoint. Thus, if u is a 0-vertex or if x starts with 0, then the found path contains at most one 1-vertex, regardless of the requested ID and of the type of the request (self-request or probe). In this case, a 1-vertex cannot learn of another 1-vertex by path overhearing. Further, we differ between the following three cases:

Case 0: u is a 0-vertex.
If the requesting vertex is a 0-vertex, then path overhearing is the only mechanism that may help with a 1-vertex learning another 1-vertex. However, we argued previously that no 1-vertex learns of another 1-vertex using path overhearing with such a request.

Case 1: u is a 1-vertex and ID x starts with 0.
Again, no 1-vertex learns of another 1-vertex using path overhearing with such a request. Further, since both V_1 and V_2 contain at least k 0-vertices and since P only consists of 0-vertices, every vertex knows at least k 0-vertices or no 1-vertices. Thus, requesting an ID that has 0 as its most significant bit always only gets 0-vertices as a response, and no 1-vertex is learned.

Case 2: u is a 1-vertex and ID x starts with 1.
We assume without loss of generalization that u is in V_1. Then, it does not know a 1-vertex in V_2 by assumption, but knows at least one other 1-vertex in V_1. Thus, the bitwise closest contact to the target is always a 1-vertex in V_1 and such a request can only be forwarded to another 1-vertex in V_1. Since, by assumption, no 1-vertex in V_1 knows a 1-vertex in V_2, no 1-vertex in V_2 is learned from the response.

In all cases, the proposed invariant holds after a request. It remains to show that the graph is indeed not KIRA-connectable. Let v be a 1-vertex in V_2 and let u be the 1-vertex in V_1 that is bitwise closest to v. Then, $B_u(v)$ is empty since no 1-vertex in V_1 knows a 1-vertex in V_2. By Lemma 3.4, G is not be KIRA-connected. ∎

We further show that the probability that G is assigned an ID distribution such that G is not KIRA-connectable is constant if we choose V_1 and V_2 large enough.

Lemma 3.6: *Let $m \in \mathbb{N}$. For $|V_1| = |V_2| = 8k + 4m$, the probability that G has an ID distribution such that G is not KIRA-connectable is at least*

$$\left(\frac{1}{2}\right)^{\ell} \cdot (1 - e^{-m})^2 \cdot \left(1 - \left(\frac{1}{2}\right)^{8k+4m+1}\right)^2.$$

Proof. For each vertex, the probability that it is a 0-vertex is $\frac{1}{2}$. As we assume that IDs are chosen uniformly at random, the probability that P consists of ℓ 0-vertices is $\left(\frac{1}{2}\right)^{\ell}$. Further, we determine the probability that there are at least k 0-vertices in V_1. Let X_i be a random variable that describes the number of 0-vertices in V_i for $i \in \{1, 2\}$. Then, the expected number of 0-vertices in V_1 is $\mathbb{E}[X_1] = |V_1| \cdot \frac{1}{2} = 4k + 2m$. Further, X_1 can be written as the sum of $|V_1|$ 0-1-valued random variables X_v for $v \in V_{i+1}$, where X_v indicates whether v is a 0-vertex. By assumption, these random variables are independently and identically distributed. Thus, we can apply Chernoff's bound (2.3) with $\alpha \coloneqq 1 - \frac{2k}{2k+m}$ to get the following lower bound for the probability:

$$\begin{aligned}\Pr[X_1 \geq k] &= 1 - \Pr[X_1 < k] = 1 - \Pr[X_1 < (1-\alpha) \cdot \mathbb{E}[X_1]] \\ &\geq 1 - \exp\left(-\alpha^2 \cdot \frac{\mathbb{E}[X_1]}{2}\right) = 1 - \exp\left(-\left(1 - \frac{2k}{2k+m} + \frac{4k^2}{(2k+m)^2}\right) \cdot (2k+m)\right) \\ &\geq 1 - \exp(-(2k+m-2k)) \ = 1 - e^{-m}.\end{aligned}$$

The event that there are at least k 0-vertices in V_2 is independent of the event that $X_1 \geq k$, and it is $\Pr[X_2 \geq k] = \Pr[X_1 \geq k]$. Further, both V_1 and V_2 have to contain at least two 1-vertices each, which is easily satisfied by choosing the sizes of V_1 and V_2 sufficiently large. More specifically, the probability that V_i for $i \in \{1, 2\}$ contains at least two 1-vertices is at least $1 - \left(\frac{1}{2}\right)^{|V_i|+1}$. Multiplying these probabilities proves the statement of the lemma. ∎

A consequence of the previous lemma is that the probability that KIRA cannot establish connectivity for such a graph G tends to $\left(\frac{1}{2}\right)^{\ell}$ for large m. Thus, this probability is constant for constant ℓ. Further, for $k \in o(m)$, the probability is asymptotically independent of k, the size of the buckets and the number of vertices that are sent back in a response. This directly implies that even if we choose k to be non-constant, for example $\log(n)$, it is still possible to construct graphs that are not KIRA connectable with constant probability.

This example can be expanded to a certain extent by using a spider graph with $c \in \mathbb{N}$ "legs" of length $\ell \in \mathbb{N}$ instead of a path and multiple independent sets $V_1, \ldots, V_c$, each connected to the endpoints of distinct legs. Again, we want the IDs to be distributed such that the ID of each vertex of the spider graph has 0 as the most significant bit and such that there are at least k 0-vertices and two 1-vertices in each independent set. With the same reasoning as for Lemma 3.5, it is impossible for a 1-vertex to learn of 1-vertices that are adjacent to another "leg". Further, if we choose the sizes of the vertex sets $V_1, \ldots, V_c$ as in Lemma 3.6, then the probability that we get an unconnectable ID distribution is

$$\left(\frac{1}{2}\right)^{c \cdot \ell} \cdot (1 - e^{-m})^c \cdot \left(1 - \left(\frac{1}{2}\right)^{8k+4m}\right)^c.$$

If c and ℓ are both constant, then this probability tends to $\left(\frac{1}{2}\right)^{c \cdot \ell}$ for large m, which is constant as well.

A natural approach to generalize this construction would be to use the constructed graph as a separator in a larger graph. For instance, it is possible to modify the vertex sets V_1 and V_2 by using arbitrary graphs, or by extending V_1 and V_2. However, such a graph is not KIRA-unconnectable in general, even if the separator receives a "bad" ID distribution as in the previous example. This is shown in Figure 3.3, where a few additional vertices are "glued" to each side of the graph. Then, it is possible that this newly obtained graph is KIRA-connectable. However, in the following section, we show that graphs that include a similar separator are still harder to connect in some sense.

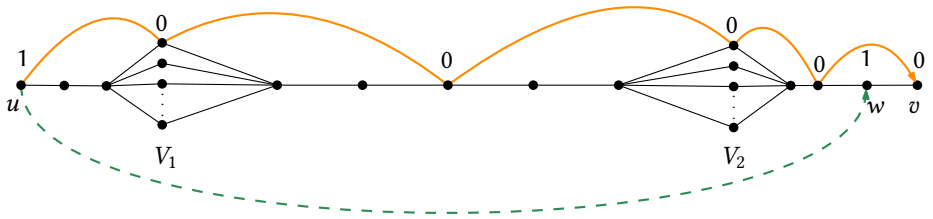

Figure 3.3: The previously unconnectable graph with two extensions on each side. Suppose that u requests $\mathrm{ID}(v)$, and KIRA finds the orange path from u to v by recursively routing to the contact that is bitwise closest to v. The underlay path contains w as a vertex, which is learned (and stored) by u using path overhearing. Thus, the two sides are KIRA-connectable, even though the subgraph separates them is not KIRA-connectable if we consider it without the extensions.

3.3 Connectivity for Different Mechanisms

Experiments with different specific graph classes suggest that KIRA is able to establish connectivity in many cases [BZDH22]. This is due to the mechanisms that KIRA implements, which help with the discovery of new contacts. In the following, we analyze how these mechanisms of KIRA contribute to establishing connectivity, and how KIRA behaves on underlay graphs with non-constant diameter.

We show that both random probing and path overhearing are needed in a way that there are graphs with certain ID distributions that cannot be KIRA-connected if the algorithm does not use both mechanisms. More specifically, we construct a graph that is assigned a "bad" ID distribution with constant probability such that KIRA is not able to establish connectivity if either random probing or path overhearing is not used. The construction uses a similar idea as the separating path of the example in Section 3.2 which is not KIRA-connectable if the ID distribution has certain properties. Unlike before, the graph has non-constant diameter and the parts that are attached to the separator are not stars.

The graph consists of two disjoint subgraphs G_1 and G_2, which are separated by a path-like structure. We will show that this separator (if assigned a certain ID distribution) prevents the discovery of underlay paths from one side of the separator to the other side if KIRA does not use random probing or path overhearing.

To prevent 1-vertices on different sides of the separator from learning from each other, we aim to have a path in the separator that only consists of 0-vertices. Analogously, the separator contains a path that only consists of 1-vertices. Formally, a *separator gadget* S consists of a path with ten vertices and two stars in the middle. We denote the first five vertices on the path by $\ell_1, \ldots, \ell_5$ and the last five vertices by $r_5, \ldots, r_1$. Further, there are two independent sets L and R which consist of k vertices each. The vertices in L are adjacent to ℓ_5 and the vertices in R are adjacent to r_5. Only the endpoints of the path ℓ_1 and r_1 are connected to other vertices in the graph. All other vertices in S do not have further edges.

We call a separator gadget *separating* if the IDs in the separator have the following property. The IDs of the vertices in $\{\ell_3, \ell_4, \ell_5\} \cup L \cup \{r_1, r_2\}$ are 0-vertices, while all other vertices of the gadgets are 1-vertices. In a separating gadget, we refer to the vertices ℓ_3, ℓ_4 and ℓ_5 and the vertices in L as the *0-block* of the separator. Analogously, the vertices r_3, r_4 and r_5 and the vertices in R form the *1-block* of the separator. Figure 3.4 shows such a separating gadget.

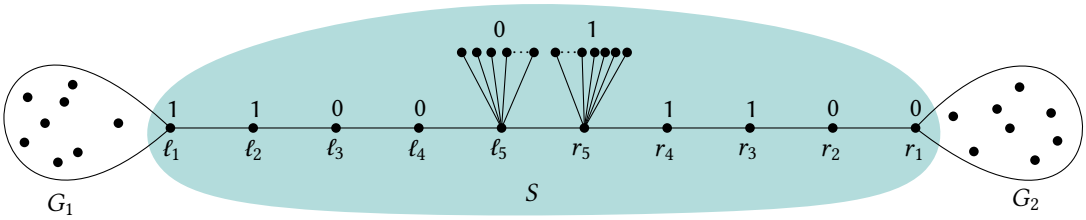

Figure 3.4: A separating gadget S, where the bits indicate the most significant bit in each ID.

For constant k, the size of the separator gadget is constant as well. Thus, if there is only one separator gadget in the graph, then the probability that it is separating is $\frac{1}{2}^{|S|}$, which is constant. To increase this probability, we form a chain of sufficiently many separator gadgets such that at least one gadget is separating with high probability. Then, the constructed graph G is a union of the subgraphs G_1, G_2 and the separators $S_1, \ldots, S_t$, where each S_i with $i \in \{1, \ldots, t\}$ forms a separator gadget. Vertices in G_1 may only be adjacent to the left endpoint of S_1 and vertices in G_2 may only be adjacent to the right endpoint of S_t. The separator gadgets are chained together in a way that the right endpoint of S_i is adjacent to the left endpoint of S_{i+1} for all $i \in \{1, \ldots, t-1\}$. Suppose that G is assigned an ID distribution such that it contains a separating gadget S^*. For better understanding, we call one side of S^* the *left* part of the graph, while the other side is the *right* part. More formally, the graph G is partitioned into two connected components by removing the edge $\{\ell_5, r_5\}$ in S^*. Then, the connected component that contains the vertex set G_1 is the left part, and analogously, the connected component containing the vertex set G_2 is the right part.

In the following, we show that a separating gadget indeed prevents vertices from one side of the separator knowing vertices on the other side if either random probing or path overhearing is not implemented. Further, we prove that G has a separating gadget with high probability for constant k.

3.3.1 Without Random Probing

First, we assume that KIRA does not use random probing while trying to establish connectivity. In other words, a vertex can only update its routing table by self-requests or by path overhearing. The following lemma shows that if there is a separating gadget S in G, then vertices in $V(G) \setminus V(S)$ on opposite sides of S do not know each other as contacts.

Lemma 3.7: *Assume that KIRA does not use random probing to establish connectivity. Let S be a separator gadget as defined above and let G be a graph with an ID distribution that has S as an induced subgraph such that S is a vertex separator in G. Assume that S is separating, and let u and v be vertices in $V(G) \setminus V(S)$. If u and v are not in the same connected component in $G \setminus S$, then u does not know v as a contact and vice versa.*

Proof. We show that the following, slightly stronger, invariant always holds if vertices only issue self-requests. Without loss of generality, let u be a vertex on the left side, and v a vertex on the right side of the graph. Note that u and v may be vertices in S. Then, u does not know v as a contact and vice versa, unless u is in the 3-hop-vicinity of v. This exception is only relevant for vertices in the 0-block and vertices in the 1-block of G.

First, we observe that with the assumption that the invariant holds, the k bitwise closest contacts of a vertex are always on the same side as the vertex itself. For vertices which are not in the 0-block or the 1-block, this property follows immediately from the invariant. As

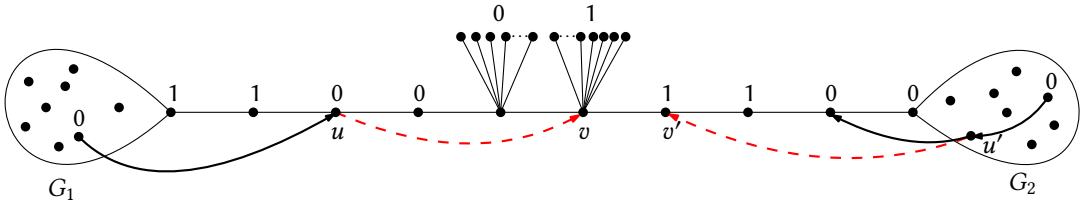

Figure 3.5: The left and the right part depict the first and second case of Lemma 3.7, respectively. In case 1, the 0-vertex u does not forward the request to the 1-vertex v. In case 2, u' does not forward the request to the 1-vertex v' since it knows another 0-vertex.

each vertex in the 0-block has at least k 0-vertices in its 3-hop-vicinity by construction, the k bitwise closest contacts of a vertex in the 0-block do not contain a 1-vertex. Since vertices of the 0-block only know 1-vertices on the right side, the k bitwise closest contacts must be on the same side. For the vertices in the 1-block, the same reasoning applies by symmetry.

Moreover, we show that self-request are only forwarded within the same side, i.e., the found path of such a request only consists of vertices of one side. Because of the skewed symmetry, self-requests of 0-vertices from the left side behave like self-requests of 1-vertices from the right side. Thus, we only consider 0-vertices issuing self-requests without loss of generalization. We treat requests from the left and from the right side in two separate cases. Both cases are depicted in Figure 3.5.

Case 1: Self-request of a 0-vertex from the left side.
Assume that the path found by the self-request contains a vertex on the right side. We argue that this is impossible if the invariant holds. Let u be the last overlay hop of the path that is on the left side of the graph and v the next overlay hop, which is on the right side. Then, u is a vertex of the 0-block, and v is a vertex of the 1-block since by assumption, vertices in the 0-block are the only vertices on the left side that know vertices on the right side. As v has another 0-vertex in its 3-hop-vicinity, the request is not forwarded to a 1-vertex since either the bitwise closest contact is chosen as the next overlay hop or the request terminates. Thus, u cannot be on the right side, which is a contradiction to our assumption.

Case 2: Self-request of a 0-vertex from the right side.
Again, we assume that the path found by the self-request contains a vertex on the left side. Then, the path contains some vertex v in the 1-block as an overlay hop since vertices in the 1-block are the only vertices that know a vertex on the left side by assumption. Let v be the first overlay hop of the path that is in the 1-block of the graph. We consider the previous overlay hop u of v. Since the request was issued by a 0-vertex, v is not the requesting vertex and thus, cannot be the first overlay hop. Then, v must be a contact of u and u knows a corresponding underlay path to v. Since v is the first overlay hop in the 1-block of the graph, such a path contains r_1 or r_2. We argue that u then knows a 0-vertex as a contact. If u is r_1 or r_2, it trivially knows another 0-vertex by discovering its 3-hop-vicinity. Otherwise, u learns of r_1 and r_2 using path overhearing, and either stores these vertices or already knows k other 0-vertices. Since a request is only sent to vertices that are bitwise closest to the target, the overlay hop after u cannot be a 1-vertex, which is a contradiction to our assumption.

As a consequence, vertices of the other side cannot be learned using path overhearing. Moreover, since the k bitwise closest contacts of a vertex are on the same side as the vertex itself, it is impossible to learn a contact from the other side from the response to the self-request. Thus, for both types of requests, the proposed invariant holds after the request which proves the statement of the lemma. ∎

This shows that there are graphs with non-constant diameter that have an ID distribution, for which KIRA cannot establish connectivity without random probing. Note that if the graph contains a separating gadget, it is impossible for KIRA without random probing to find a path from the left to the right side of the separator, no matter which requests are issued in which order.

3.3.2 Without Path Overhearing

Now, we assume that KIRA uses random probing and self-requests but not path overhearing. In this case, we need an additional condition to make the separator work. We further assume that every vertex knows at least k vertices on the same side that have the same most significant bit as the vertex itself. This is the case if the k closest contacts of a vertex are not in the bucket of depth 0. Based on the Lemma 3.7, we obtain a similar statement if KIRA does not use path overhearing.

Lemma 3.8: *Assume that KIRA does not use path overhearing to establish connectivity. Let S be a separator gadget as defined above and let G be a graph that has S as an induced subgraph such that S is a vertex separator in G. Further, assume that every vertex knows at least k vertices with the same most significant bit on the same side of the separator. Assume that S is separating, and let u and v be vertices in $V(G) \setminus V(S)$ with the same most significant bit. If u and v are not in the same connected component in $G \setminus S$, then u does not know v as a contact and vice versa.*

Proof. We show that the following invariant holds. Let u be a vertex on the left side and v a vertex on the right side of the graph such that u and v share the same most significant bit. Then, u does not know v as a contact and vice versa. Note that the invariant shown in Lemma 3.7 is similar, but slightly stronger since there, u and v do not necessarily share the most significant bit.

Again, we differ between different types of requests. By assumption, the k closest contacts of each vertex are on the same side as the vertex itself. Thus, for self-requests, we can argue exactly as in the proof of Lemma 3.7. For requests from vertices in $V(G) \setminus S$, the same holds as for self-requests of those vertices, and they do not cross to the other side. The only type to be considered are requests from vertices in the 0-block and vertices from the 1-block. Probes for 0-vertices from the 0-block can be treated just as self-requests. The same holds for probes for 1-vertices from the 1-block.

If a vertex in the 0-block requests a 1-vertex, this request may be forwarded to a vertex in the 1-block, and the request terminates at some 1-vertex v on the right side of the separator. As we assume that v knows at least k 1-vertices on the right side of the separator, v only returns 1-vertices, and the requesting vertex does not learn of a 0-vertex on the other side.

The case that a vertex in the 1-block requests a 0-vertex is symmetric. ∎

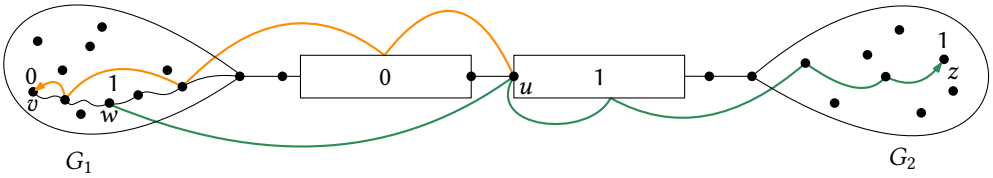

Figure 3.6: An example graph with an ID assignment such that both sides of a separating gadget are KIRA-connectable if random probing and path overhearing are used. The boxes represent the 0-block and the 1-block. Suppose that a vertex u in the 1-block requests a 0-vertex v, and KIRA finds a path from u to v. The underlay path contains w as a vertex, which learns of u using path overhearing. A request from w to z may skip the 0-block now, routing directly to u and from there to z. Thus, a 1-vertex on one side of the separating gadget may be KIRA-connected to a 1-vertex on the other side.

3.3.3 Probability of a Separating Gadget

In the following, we show that it is likely that G contains a separating gadget if $n := |V(G)|$ is sufficiently large. For this, we choose the sizes of G_1 and G_2 and the parameter t depending on k and the total number of vertices n such that the probability that there is a separating gadget in G is high, while the graphs G_1 and G_2 contain not too few vertices.

Lemma 3.9: *Let G be defined as above with $m := |V(G_1)| + |V(G_2)| = n^{\frac{2}{3}}$ and $t = \frac{n-m}{2k+10}$. Then, G contains a separating gadget with high probability.*

Proof. We first determine the probability that a separator gadget is separating. By definition, a separator gadget S is separating if the vertices in $\{\ell_1, \ell_2, \ell_3, \ell_4\} \cup U$ are 0-vertices and if the other vertices in S are 1-vertices. Since we assume that IDs are distributed uniformly at random, the probability that a single vertex is a b-vertex for some $b \in \{0, 1\}$ is $\frac{1}{2}$. Then, the probability that a single separator gadget is separating is $\left(\frac{1}{2}\right)^{|S|} = \left(\frac{1}{2}\right)^{2k+10}$. With t separator gadgets, the probability that there is at least one separating gadget is

$$\begin{aligned}
1 - \left(1 - \left(\frac{1}{2}\right)^{2k+10}\right)^t &= 1 - \left(1 - \left(\frac{1}{2}\right)^{2k+10}\right)^{\frac{n-n^{\frac{2}{3}}}{2k+10}} \\
&\geq 1 - \exp\left(-2^{-2k-10}\right)^{\frac{n-n^{\frac{2}{3}}}{2k+10}} \\
&= 1 - \exp\left(-\frac{n - n^{\frac{2}{3}}}{(2k+10) \cdot 2^{2k+10}}\right)
\end{aligned}$$

by using the well-known inequality $1 - x \leq \exp(-x)$ for all $x \in \mathbb{R}$. Since we assume that k is a constant, this proves the statement from the lemma. ∎

Note that if we increase k to $\log(n)$, the probability that G contains a separating gadget is only constant. Thus, it may be reasonable not to choose k as a constant in the algorithm but to let it depend on the size of the graph.

3.3.4 With Path Overhearing and Random Probing

If KIRA implements both path overhearing and random probing, then there exist instances with a separating gadget, where it is possible that a vertex on one side of the separator knows a vertex on the other side of the separator. In this case, the invariants from Lemma 3.7 and

Lemma 3.8 do not hold. Figure 3.6 shows an example of such an instance. As we have seen in two examples (Figure 3.3 and Figure 3.6), it is not sufficient for separators to have "bad" ID assignments. It seems that ID assignments need to satisfy global conditions (instead of only local ones) such that paths as in Figure 3.6 cannot be found by KIRA. Intuitively, if there is a large number of paths in a graph that contain two vertices u and v, then it is more likely that some other path is routed via these two vertices, and that one vertex learns of the other by path overhearing. This leads us to the following conjecture.

Conjecture 3.10: *Let G be a graph, and suppose that every pair of vertices is contained in a sufficiently large number of distinct paths in G. Then, G is KIRA-connectable with high probability.*

3.4 Establishing KIRA-Connectivity

In this section, we consider slightly modified versions of KIRA, and prove that these versions are able to establish KIRA-connectivity in specific scenarios. First, we show that if a new vertex joins a KIRA-connected graph, then KIRA can establish KIRA-connectivity for the new graph as well. Then, we prove that a very specific version of KIRA is always able to establish connectivity. Note that this specific version is not applicable in practice. Further, we study KIRA-connectivity specifically on paths.

3.4.1 Join New Vertex to Connected Graph

In the following, we assume that a graph G is already KIRA-connected. We show that a vertex v that joins the network can be connected to the rest of the graph by at most B requests, where B is the number of bits in the ID space, if we modify the algorithm slightly as follows. Vertices are allowed to probe deterministically (without global knowledge of the graph), i.e., vertices can request paths to specific IDs. Particularly, a vertex requests IDs in distance ranges in which the vertex does not know any contact. With the original algorithm, such a request might immediately be discarded if it cannot be forwarded to a contact that improves the bit distance to the target. We assume that in this case, via-probing is used, i.e., the requesting vertex can choose a contact which is supposed to be the first overlay hop. Further, vertices use path overhearing to add and update contacts.

For the newly added vertex v, we describe the following strategy that results in the new graph being KIRA-connected. The vertex v starts with discovering the 3-hop-vicinity. Then, it repeatedly requests IDs that belong to buckets which are still empty, starting with the deepest bucket and its own ID. More specifically, it requests IDs that have minimal bit distance to v for each empty bucket. The first overlay hop for the via-probe is chosen arbitrarily among its contacts by v for each request. With the responses and the corresponding paths, the vertex fills its routing table. We show in the following lemma that after probing an ID of every empty bucket once, the graph is KIRA-connected again.

Lemma 3.11: *Let G be a graph that is already KIRA-connected. Assume that a new vertex v with empty routing tables is added to the graph. Using deterministic via-probing, there is a strategy for v to request IDs such that the new graph is KIRA-connected after at most B requests.*

Proof. Let bucket of depth $i \in \{0, \dots, B-1\}$ be the deepest bucket in the routing table of v that is empty after discovering the 3-hop-vicinity. The vertex then requests an ID x that belongs to this bucket and that has minimal bit distance to v, i.e., x is obtained from $\mathrm{ID}(v)$ by flipping the i-th bit. This request is forwarded to a contact u of v. Since the graph is KIRA-connected,

the algorithm finds a path from u to the vertex whose ID is bitwise closest to the requested ID, which we denote by w. If w belongs in the empty bucket $B_v(x)$, then v stores w in $B_v(x)$. Otherwise, there is no vertex in the graph that belongs to $B_v(x)$. In this case, the bucket stays empty. After repeating this for every empty bucket, $B_v(u) \neq \emptyset$ holds for every vertex $u \in V(G)$, i.e., every bucket of v contains a vertex if the corresponding bit distance range is not empty. By Lemma 3.2, $G \cup \{v\}$ is then KIRA-connected.

Further, we argue that the algorithm finds a path from every vertex in the graph to v. Since v attempted to fill every bucket of its routing table, it knows its bitwise closest vertex in G. This vertex also stores v in a bucket since v is the only vertex that belongs to that range. Otherwise, there would be a vertex that is even closer to v. Now, let u be some vertex that requests $\mathrm{ID}(v)$. Either, u knows v as a contact, in which case it directly routes to v, or it is forwarded to a vertex that is bitwise closer to v. Since G is KIRA-connected, such a contact must exist if the current vertex is not the bitwise closest. If it is the bitwise closest vertex, it directly routes to v. ∎

However, generalizing this approach for multiple KIRA-connected components that are connected via one vertex does not work. In particular, let G be a graph and $v \in V(G)$ a vertex such that $G \setminus \{v\}$ has two connected components V_1 and V_2. Suppose that both V_1 and V_2 are KIRA-connected, but no vertex in V_1 knows a vertex in V_2 and vice versa. In other words, V_1 and V_2 are two disjoint KIRA-connected components. In this case, requests from v may not be sufficient to establish KIRA-connectivity for G. More specifically, if v requests an ID in V_2 but routes to a vertex in V_1 first, it is not able to find the requested ID.

3.4.2 With Deterministic Via-Probing and Propagation

In this subsection, we consider a further modified algorithm with deterministic via-probing, propagation, and path overhearing. In particular, every vertex is able to request a specific ID and choose a specific via-contact to be the first overlay hop. Further, we modify the replacement strategy of contacts in full buckets. In the original algorithm, contacts are replaced using the Proximity Neighbor Selection policy, where contacts with the shorter corresponding underlay path are preferred. This may result in a local optimum, where contents in buckets do not change anymore, but the graph is not KIRA-connected. To prevent this, we use propagation instead of a replacement strategy.

We show that with the modified algorithm, it is always possible for KIRA to establish connectivity, no matter the current state of the routing tables. We say that a vertex u has *seen* some vertex v if a request was issued that was routed via both u and v. Note that since the order of the vertices does not matter, this relation is symmetric. Since KIRA employs path overhearing, this is equivalent to u knowing an underlay path to v and vice versa. First, we observe a useful property if propagation is used instead of a replacement strategy.

Lemma 3.12: *Assume propagation is used instead of a replacement strategy for some graph G. Then, the following holds for all vertices u and v in $V(G)$: If u has seen v, then u is KIRA-connected to v. Moreover, u stays KIRA-connected to v, regardless of requests issued in the future.*

Proof. Assume some vertex v has seen u. If $B_v(u)$ is not full, then v stores u as a contact and is KIRA-connected to u. Since contacts are not replaced in the modified algorithm, future requests do not affect the KIRA-connectivity from v to u. On the other hand, if $B_v(u)$ is full, then u is propagated until it reaches a vertex w, where $B_w(u)$ is not full, and w stores u as a

contact. Since no replacement strategy is used, buckets that are already full never change. Thus, if v requests u, the request is routed to the same vertices as u was propagated until it reaches vertex w, which knows u as a contact. In both cases, v is KIRA-connected to u. ∎

Using the original algorithm, KIRA-connectivity can be destroyed between two vertices if new contacts are learned by Lemma 3.1. However, as a consequence of the previous lemma, propagation guarantees that once a vertex has seen another vertex, KIRA can always find a path between these vertices. The following lemma makes use of this property by iteratively expanding the set of vertices that a vertex has already seen for every vertex in the graph.

Lemma 3.13: *Assume propagation and deterministic via-probing are used for some graph G. Then, KIRA is always able to establish KIRA-connectivity.*

Proof. For some vertex v, consider the set of vertices $s(v)$ that vertex v has already seen. Initially, this set only contains vertices in the 3-hop-vicinity of v. By Lemma 3.12, all vertices in $s(v)$ are KIRA-connected to v and also stay so. Assume $s(v)$ does not consist of all vertices in $V(G)$. Since the overlay graph is strongly connected, there is a vertex $w \notin s(v)$ that has a contact $u \in s(v)$. This is depicted in Figure 3.7. Vertex w can now successfully request v via its contact u and is then added to $s(v)$. This is repeated until $s(v) = V(G)$ for all vertices $v \in V(G)$. Then, G is KIRA-connected. ∎

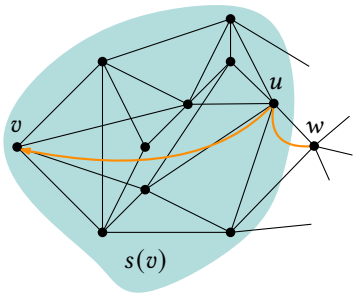

Figure 3.7: The situation as described in the proof.

Note that this is just a theoretical concept and is not applicable in practice since using propagation instead of a replacement strategy creates an extremely large overhead of requests. Further, vertices need to request very specific IDs for KIRA-connectivity to be established. However, vertices do not have a global view on the graph and have no knowledge of which IDs they should request, thus, the probability that these specific IDs are requested is small.

3.4.3 KIRA-Connectivity on Paths

In this section, we analyze KIRA-connectivity specifically on paths. We propose a deterministic strategy using (partly modified) mechanisms implemented in KIRA that establishes KIRA-connectivity if the ID assignment satisfies certain constraints. Further, we show that these constraints are satisfied with high probability.

We first introduce some notation that we use in the following. Let $\mathcal{G}$ be the overlay graph of some graph G. There is an edge from u to v in $\mathcal{G}$ if u knows v as a contact. Recall that each edge in the overlay graph corresponds to a unique path in the underlay graph. We define a *prefix graph* $\mathcal{G}[\mathrm{p}]$ for some bit string $\mathrm{p} \in \{0,1\}^*$ with length $|\mathrm{p}|$ to be the induced subgraph of $\mathcal{G}$ that consists of all vertices in $\mathcal{G}$, whose IDs start with p. More formally, it is $V(\mathcal{G}[\mathrm{p}]) := \{v \in V(\mathcal{G}) \mid v \text{ is a p-vertex}\}$. Note that $\mathcal{G}[\varepsilon]$ is isomorphic to $\mathcal{G}$, where ε is the empty bit string.

For a graph $\mathcal{G}[\mathrm{p}]$, we introduce the notion of p-connectivity. A vertex $u \in V(\mathcal{G}[\mathrm{p}])$ is p-connected to a vertex $v \in V(\mathcal{G}[\mathrm{p}])$ if there is a path in $\mathcal{G}[\mathrm{p}]$ from u to v. Further, we define p-*components* that partition the vertices of $\mathcal{G}[\mathrm{p}]$ into p0-components and p1-components. Two p0-vertices u and v are in the same p0-component if and only if they are p0-connected and if the corresponding underlay path does not contain a p1-vertex. Analogously, two p1-vertices u and v are in the same p1-component if and only if they are p1-connected and if the

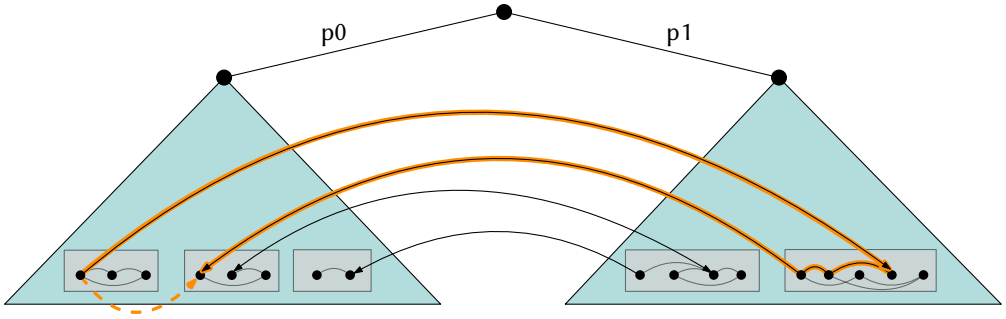

Figure 3.8: An example for a prefix graph $\mathcal{G}[\mathrm{p}]$. The gray boxes represent p0-components and p1-components. Vertices that are incident to edges that connect two vertices of different subtries are border vertices. If the dashed orange edge between two p1-components is added, then it acts as a bridge. We obtain the corresponding underlay path of the orange edge by concatenating the corresponding underlay paths of the edges in $\mathcal{G}[\mathrm{p}]$ that are highlighted in orange.

corresponding underlay path does not contain a p0-vertex. A *border vertex* of a p0-component V_0 of $\mathcal{G}[\mathrm{p}]$ is a vertex in V_0 that knows a p1-vertex in some p1-component V_1 as a contact with a corresponding underlay path that does not contain any p0-vertices as inner vertices. In this case, we say that V_0 and V_1 are *adjacent* components. A newly added edge in $\mathcal{G}[\mathrm{p}]$ is a *bridge* if it connects two pb-components with $\mathrm{b} \in \{0, 1\}$ and if the corresponding underlay path does not contain any pb-vertices as inner vertices. In this case, the two components are merged into one. Figure 3.8 depicts an example of a prefix graph $\mathcal{G}[\mathrm{p}]$. If the underlay graph G is a path, we say that two p-vertices u and v with $\mathrm{b} \in \{0, 1\}$ are *p-consecutive* if the path from u to v in G contains no p-vertices as inner vertices.

Modified KIRA We propose a strategy that uses modified mechanisms of KIRA to establish KIRA-connectivity on a path. Let G be a graph and $\mathcal{G}$ its initial overlay graph. We show that this algorithm is able to establish connectivity for a graph G if there are at most $k - 1$ p-consecutive p0-vertices and at most $k - 1$ p-consecutive p1-vertices in G for every prefix p. For simplicity, we assume that only the 1-hop-vicinity is discovered initially, i.e., the topology of the initial overlay graph $\mathcal{G}$ is the same as G.

The idea is to consider $\mathcal{G}[\mathrm{p}]$ for all prefixes successively, sorted increasingly by length. For each $\mathcal{G}[\mathrm{p}]$, we want to bridge over each p0- and p1-component such that $\mathcal{G}[\mathrm{p0}]$ and $\mathcal{G}[\mathrm{p1}]$ are connected. We refer to the current length of the prefix p as the current *level* ℓ. First, we assume that each vertex in the network knows the value of ℓ. However, this is not possible in practice, and we later argue that the algorithm can be modified such that each vertex v stores its own level ℓ_v.

For each $\mathcal{G}[\mathrm{p}]$ of some level ℓ, vertices only request p-vertices. In other words, requests in $\mathcal{G}[\mathrm{p}]$ stay within $\mathcal{G}[\mathrm{p}]$ in the sense that all visited overlay hops are in $\mathcal{G}[\mathrm{p}]$. As a consequence, we can consider each $\mathcal{G}[\mathrm{p}]$ independently of each other (assuming that other requests are forwarded in the underlay but ignored otherwise). Suppose now that $\mathcal{G}[\mathrm{p}]$ is connected. In the ID trie, the vertices of $\mathcal{G}[\mathrm{p}]$ are located in some subtree T whose root has depth ℓ. This subtree can be partitioned into two subtries T_0 and T_1, which contain vertices with the prefix p0 and p1, respectively. The graphs $\mathcal{G}[\mathrm{p0}]$ and $\mathcal{G}[\mathrm{p1}]$ that correspond to T_0 and T_1, respectively, might not be connected, and one goal of the algorithm is to make each graph connected. Further,

each vertex in $\mathcal{G}[\mathrm{p}0]$ should know a vertex in $\mathcal{G}[\mathrm{p}1]$ and vice versa after level ℓ. Then, the same procedure is repeated for the subtries in level $\ell + 1$. By Lemma 3.2, the graph is then KIRA-connected.

In the following, we describe the mechanisms that we partly modified. Requests are always deterministic probes (without global knowledge) and stored vertices are never replaced. Let ℓ be the current level. The main difference is that newly learned vertices (by path overhearing or by closest-k-response) are only stored under certain conditions, which depend on the current level. Suppose that vertex $v \in \mathcal{G}[\mathrm{p}]$ for some p learns of a vertex u. If $B_v(u)$ has depth at most ℓ, then v stores u as a contact if $B_v(u)$ is not full yet. The intuition behind this is that vertices in lower depths do not matter to the algorithm anymore and thus, can be stored arbitrarily. If $B_v(u)$ has depth more than ℓ, then v stores u if and only if the corresponding edge is a bridge in $\mathcal{G}[\mathrm{p}]$.

We denote the state of the overlay graph after level ℓ by $\mathcal{G}_\ell$. Let ℓ be the current level and p a bit string with length ℓ. Let $V_0 \subseteq \mathcal{G}[\mathrm{p}]$ be the set of border vertices of p0-components in $\mathcal{G}[\mathrm{p}]$. By definition, each border vertex $v \in V_0$ knows at least one p1-vertex u as a contact with a corresponding path that does not contain any p0-vertices as inner vertices. The vertex v then requests vertex u. If it stores a newly learned p0-vertex, then v does not send further requests during level ℓ. Otherwise, it repeatedly sends requests for p1-contacts with a corresponding underlay path without p0-vertices as inner vertices that have not been requested yet. We show that in this case, a bridge is eventually found. Once all vertices in $V(\mathcal{G})$ stopped sending requests, the level ℓ is increased by one, and $\mathcal{G}_{\ell+1}$ is set to the last state of $\mathcal{G}_\ell$.

We show in the following that G is KIRA-connected after level B, where B is the number of bits in the ID space. For this, we first prove two useful invariants for the overlay graph after each level.

Lemma 3.14: *Let $\mathcal{G}_\ell$ be the overlay graph before level ℓ for some $\ell \in \{0, \dots, B-1\}$. Suppose that $\mathcal{G}_\ell$ has the following properties:*

1. *For every prefix p of length ℓ, the prefix graph $\mathcal{G}_\ell[p]$ is a path where the vertices are in the same order as in the underlying path. Further, each component in $\mathcal{G}_\ell[p]$ has size at most $k-1$.*
2. *Each vertex has a contact of depth $\ell - 1$ if $\ell > 0$, and if a vertex that belongs in this bucket exists.*

Then, after level ℓ, the graph $\mathcal{G}_{\ell+1}$ has the same properties (with $\ell + 1$ instead of ℓ).

Proof. Suppose $\mathcal{G}_\ell$ has the properties as stated in the lemma before level ℓ. This situation is shown in Figure 3.9 for $\mathcal{G}_\ell[\mathrm{p}]$ for some prefix p. We show that the graphs $\mathcal{G}_{\ell+1}[\mathrm{p}0]$ and $\mathcal{G}_{\ell+1}[\mathrm{p}1]$ form a path for every prefix p of length ℓ. As mentioned previously, each $\mathcal{G}_\ell[\mathrm{p}]$ can be analyzed independently of each other.

Assume that u is a border vertex of some p0-component V_0 in $\mathcal{G}_\ell[\mathrm{p}]$ and that u knows a border vertex v of an adjacent p1-component V_1. Then, it repeatedly requests p1-vertices which have not been requested yet. We prove that it finds a bridge, i.e., a p0-vertex with a corresponding underlay path that contains no p0-vertices as inner vertices if such a vertex exists. If V_1 is adjacent to another p0-component $V_0' \neq V_0$, then there exists a border vertex $w \in V_1$ that knows a p0-contact $z \in V_0'$. By repeatedly requesting p1-vertices, u learns and stores all vertices in V_1 since V_1 consists of at most $k-1$ vertices. In particular, u requests vertex w at some point, which responds with the k closest contacts it knows. Since w only knows two contacts in $\mathcal{G}_\ell[\mathrm{p}]$, the response contains $z \in V_0'$. This vertex is then stored by u

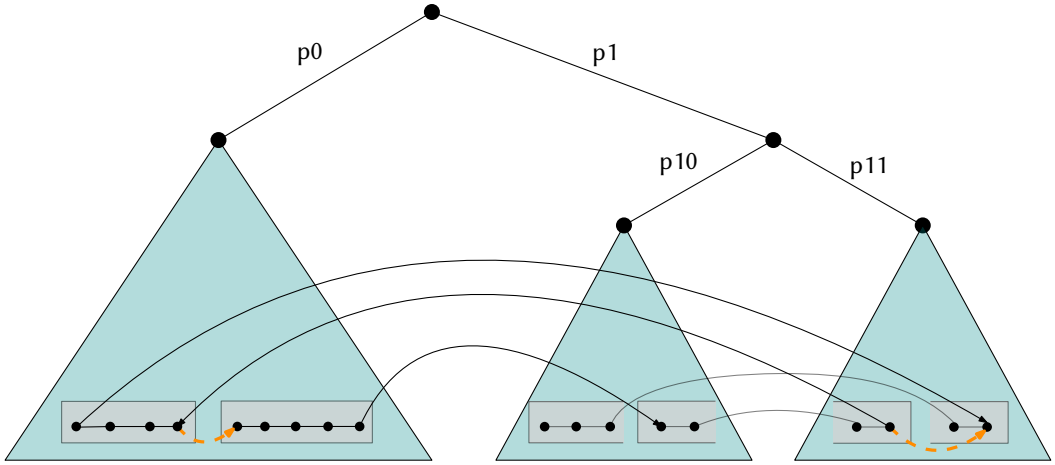

Figure 3.9: A possible overlay graph $\mathcal{G}_\ell[\mathrm{p}]$ for $|\mathrm{p}| = \ell$ that satisfies the invariants of Lemma 3.14. The gray boxes represent p0-components and p1-components. Since we do not differ between p10-vertices and p11-vertices in $\mathcal{G}_\ell[\mathrm{p}]$, the graph contains exactly two p1-components. The vertices in $\mathcal{G}_\ell[\mathrm{p}]$ form a path. In the next step, the bridges colored in orange are added to the graph. Note that both $\mathcal{G}_{\ell+1}[\mathrm{p0}]$ and $\mathcal{G}_{\ell+1}[\mathrm{p1}]$ are connected then. Then, we use induction on both subgraphs. In particular, $\mathcal{G}_{\ell+1}[\mathrm{p1}]$ consists of two p10-components and of one p11-component.

since the underlay path from u to z does not contain any p0-vertices as inner vertices. Further, vertices in V_1 learn of v by path overhearing and store v as a contact. If this is repeated for all border vertices in $\mathcal{G}_\ell[\mathrm{p}]$, then the p0-components are connected in $\mathcal{G}_{\ell+1}[\mathrm{p0}]$ and each p0-vertex knows at most two other p0-vertices which are closest to it in the underlay graph. The symmetric result holds for p1-vertices. Thus, the first part of the lemma also holds for $\mathcal{G}_{\ell+1}$. Since each p0-vertex learns of a p1-vertex and vice versa, the second part of the lemma holds for $\mathcal{G}_{\ell+1}$ as well. ∎

Theorem 3.15: *After level B, the graph is KIRA-connected.*

Proof. By repeatedly applying Lemma 3.14 to each level, each vertex has at least one contact in each bucket if such a contact exists after level B. Then, the graph is KIRA-connected by Lemma 3.2. ∎

In the previous proof, ℓ was treated as a global variable, for which all vertices know the current value. However, this is not applicable in practice. If every vertex simply stores its own level, it is possible that vertices that still have a lower level (i.e., they have may not have filled deeper buckets) cannot find a bridge that goes over a vertex in a higher level. This can be solved by ensuring that the level difference is not too high locally. In particular, a vertex in the underlying graph only forwards a request if the level of the requesting vertex is at most its own level. Otherwise, the request simply terminates.

Further, note that the number of requests needed for each level can be improved by storing newly learned vertices less restrictively. Suppose v learns of a new vertex u in level ℓ. If $B_v(u)$ has depth ℓ, we assume without loss of generality that v is a p0-vertex and u is a p1-vertex. Then, vertex u is stored if $B_v(u)$ is empty or if the corresponding underlay path from v to u does not contain a p0-vertex as an inner vertex. If $B_v(u)$ has depth more than ℓ, we assume without loss of generality that both v and u are p0-vertices. Then, vertex u is stored if $B_v(u)$ is

empty and if the number of contacts of v with depth more than ℓ is at most $k-2$ afterwards. Additionally, u is also stored by v if the corresponding edge is a bridge. As a consequence, the graph $\mathcal{G}_\ell[\mathrm{p}]$ for some p with length ℓ is not necessarily a path. Thus, border vertices may find a bridge faster between two p0-components by not having to request all vertices in the p1-component in between. However, we have to ensure that each component in $\mathcal{G}_\ell[\mathrm{p}]$ for every prefix p and every level ℓ is still sufficiently small. To prove a similar result as Lemma 3.14, more invariants are required.

Probability of Constraints In the previous part, we assumed there are at most $k-1$ p-consecutive p0-vertices and at most $k-1$ p-consecutive p1-vertices in $\mathcal{G}[\mathrm{p}]$ for every prefix p. We call the event that this condition on the ID assignment is satisfied A, and prove that the probability A occurs is high. In the following, we first consider the event that this condition does not hold for each prefix separately, and then combine these results to obtain an upper bound for the probability of $\overline{A}$, the complementary event to A.

For a fixed prefix p, this is equivalent to flipping a fair coin $|V(G[\mathrm{p}])|$ times and considering the longest streak of the same bit, which has already been extensively studied in the past [ER75 | Sch90]. For easier notation, we denote the number $|V(G[\mathrm{p}])|$ of p-vertices in G by n_p. As IDs are assigned randomly, n_p is a random variable. Let E_p be the event that there are more than $k-1$ p-consecutive p0-vertices or p1-vertices in G. The following lemma determines an upper bound for E_p given n_p (see [MU05], exercise 1.09).

Lemma 3.16: *For a prefix p, the probability that E_p occurs given n_p is*

$$\Pr\left[E_p \mid n_p\right] = \frac{n_p}{2^{k-1}}.$$

Proof. We enumerate the p-vertices in G canonically from 1 to n_p, as they are ordered in G. The probability that a streak of k p-consecutive vertices starts at the i-th p-vertex for some $i \in \{1, \ldots, n_\mathrm{p} - k\}$ is $\frac{1}{2} \cdot \frac{1}{2^k}$. We denote this event by E_p^i. Then, it is

$$\Pr\left[E_\mathrm{p} \mid n_\mathrm{p}\right] = \Pr\left[E_\mathrm{p}^1 \cup E_\mathrm{p}^2 \cup \ldots \cup E_\mathrm{p}^{n_\mathrm{p}-k} \;\middle|\; n_\mathrm{p}\right] \leq \sum_{i=1}^{n_\mathrm{p}-k} 2^{-k+1} \leq \frac{n_\mathrm{p}}{2^{k-1}}.$$

The upper bound is obtained using the union bound (2.1). ∎

We can now write the event $\overline{A}$ as a union of all E_p given n_p, and determine an upper bound for the probability of $\overline{A}$, which gives us a lower bound for the probability of A.

Lemma 3.17: *The probability that A occurs is*

$$\Pr[A] = 1 - \frac{n \cdot B}{2^{k-1}}.$$

Proof. It is $\overline{A} = \bigcup_\mathrm{p} E_\mathrm{p}$, and we again use the union bound (2.1) to obtain an upper bound for $\Pr[A]$. Then, it is

$$\Pr\left[\overline{A}\right] = \Pr\left[\bigcup_\mathrm{p} E_\mathrm{p} \;\middle|\; n_\mathrm{p} \text{ for all prefixes p}\right] \leq \sum_\mathrm{p} \Pr\left[E_\mathrm{p} \mid n_\mathrm{p}\right].$$

By sorting the summands by the length of the prefix, we rearrange the sum, and then, apply the result from Lemma 3.16.

$$= \sum_{i=0}^{B-1} \sum_{\substack{\mathsf{p} \\ |\mathsf{p}|=i}} \Pr\left[E_{\mathsf{p}} \,\middle|\, n_{\mathsf{p}}\right] \leq \sum_{i=0}^{B-1} \sum_{\substack{\mathsf{p} \\ |\mathsf{p}|=i}} \frac{n_{\mathsf{p}}}{2^{k-1}}.$$

Since the vertices in $V(G)$ are partitioned using prefixes of length i, the sum of all n_{p} with $|\mathsf{p}| = i$ for some $i \in \{1, \ldots, B-1\}$ is the total number of vertices. This gives us

$$= \sum_{i=0}^{B-1} \frac{n}{2^{k-1}} = \frac{n \cdot B}{2^{k-1}}.$$

Then, the probability of A is

$$\Pr[A] = 1 - \Pr\left[\overline{A}\right] \geq 1 - \frac{n \cdot B}{2^{k-1}}.$$

■

We assume that the ID space has size $N \coloneqq n^c$ for some constant $c > 2$ (see Lemma 2.2), i.e., the number of bits B is $c \log(n)$. Let $k \geq 2 \cdot \log(n)$ be the size of the buckets. Then,

$$\Pr[A] \geq 1 - \frac{n \cdot B}{2^{k-1}} \geq 1 - \frac{2c \cdot n \log(n)}{n^2} = 1 - \frac{2c \log(n)}{n} \to 1 \text{ for } n \to \infty.$$

Thus, the desired conditions for the ID assignment are satisfied with high probability.

Generalization A natural approach for arbitrary graphs is trying to generalize the proposed strategy for paths. Most ideas of the strategy on paths can be adopted for graphs in general, but the way of storing new contacts needs to be adapted since some vertices might need to find multiple bridges in the same prefix graph (contrary to only having to find one bridge for paths). A key difference between arbitrary graphs and paths is that vertices in $\mathcal{G}[\mathsf{p}]$ might have high degree, even if the constraint that components in $\mathcal{G}[\mathsf{p}]$ have small size is still in place. This is particularly problematic for border vertices of components. For instance, suppose that some border vertex v of a p0-component in $\mathcal{G}[\mathsf{p}]$ knows $k-1$ p0-vertices and $k-1$ p1-vertices that belong to distinct components. Then, v always returns all p0-vertices and only a small subset of the p1-vertices in a k-closest response. This may prevent p1-vertices from finding necessary bridges to make $\mathcal{G}[\mathsf{p1}]$ connected.

This may be solved by changing the way vertices select contacts for their responses such that they do not necessarily respond with the closest contacts. Instead, it might be reasonable to diversify the response by including contacts from different buckets, possibly even depending on the ID of the requesting source. Especially in the earlier stages of the algorithm, for example when the graph is not KIRA-connected yet, such a change might help with discovering contacts in other parts of the graph and escaping local optima.

4 Stretch of Found Paths

To route efficiently in large networks, it is crucial to keep the lengths of the found paths as short as possible. For other ID-based routing protocols such as VRR and DISCO, the quality of the found paths was not prioritized [Cae+06 | Sin+10]. In contrast, KIRA uses certain policies such as Proximity Neighbor Selection and Proximity Routing to find more efficient paths. On the evaluated graphs, KIRA indeed often finds reasonably short paths, especially in a later phase after a certain number of requests [BZDH22]. In the following, we analyze how long a path between two vertices found by KIRA can be if Proximity Neighborhood Selection and Proximity Routing are used. For two vertices s and t of a graph, we call the ratio of the lengths of the found path P and the shortest underlay path from s to t the *path stretch* $S_{s,t} = \frac{|P|}{\text{dist}(s,t)}$.

To evaluate the stretch, we make certain assumptions on the state of the routing tables, which we state in the following section. The rest of the chapter is divided into two parts, where we provide lower bounds for the stretch for deterministic ID assignments first and then, for random ID assignments. For deterministic ID assignments, we additionally show that the bound is tight.

4.1 Assumed State of Overlay Graph

Especially at the start-up of KIRA, the stretch of a path found by KIRA may be very high if it is found at all. However, by adding new contacts and improving existing ones, the stretch decreases over time and the KIRA-connectivity improves. Thus, it is reasonable to only consider states of the overlay graph, where KIRA-connectivity is already established, and where routing tables are already filled sufficiently well. More specifically, we provide lower bounds for the stretch if we assume a best possible state of the overlay graph regarding Proximity Neighborhood Selection.

In particular, we assume that the buckets of each vertex are filled as much as possible. This means that a bucket contains exactly k vertices, or it stores all vertices in the graph that belong to this bucket. An overlay graph whose routing tables have this property is called a *full overlay graph*. This assumption guarantees that the graph is KIRA-connected by Lemma 3.2, i.e., the algorithm finds a path between any two vertices. Moreover, the prefix bit distance to some target decreases at each hop, thus, we only need the XOR metric as a tiebreaker if Proximity Routing yields multiple candidates.

Further, we assume that the k contacts that are stored in each bucket of some vertex u are the vertices in the corresponding bit distance range that are closest to v in the underlay graph. In other words, if some vertex u is a contact of v, then there is no vertex w with $B_v(w) = B_v(u)$ and $\text{dist}(v, w) < \text{dist}(v, u)$ that is not a contact of v. We use the XOR metric as a tiebreaker, i.e., if there are multiple vertices that belong to a bucket with the same underlay distance to v, the vertex that is bitwise closest to v is preferred. Additionally, we want the underlay paths stored for each contact to be shortest paths between the vertices in the underlay graph. We call a full overlay graph with these properties an *optimal overlay graph* regarding Proximity Neighborhood Selection. Such an overlay graph is optimal in the sense that routing tables do not change anymore, and contacts are never replaced.

4.2 Worst Case ID Assignment

We investigate the worst case for the stretch using deterministic IDs, i.e., how large the stretch can be if we choose a worst case underlay graph and an ID assignment. For two vertices s and t with underlay distance $\text{dist}(s,t)$, a trivial upper bound for the stretch is $S_{s,t} \leq \frac{n-1}{\text{dist}(s,t)}$, where n is the number of vertices in the graph. The following lemma bounds the possible stretch of the path found by KIRA between two vertices, depending on their prefix bit distance.

Lemma 4.1: *Let G be a graph with a full overlay, and let $s, t \in V(G)$ be two vertices with prefix bit distance $\text{dist}_p(s,t)$. Then, the stretch $S_{s,t}$ is at most $2^{\text{dist}_p(s,t)} - 1$.*

Proof. As we assume that the overlay graph is full, a path from s to t found by KIRA uses at most $\text{dist}_p(s,t)$ overlay hops by Corollary 3.3. Let $P = (v_0 = s, v_1, \ldots, v_h = t)$ for some $h \leq \text{dist}_p(s,t)$ be the path that is found by KIRA in the overlay graph from s to t. Suppose that the request is at the i-th overlay hop v_i.

We first show that $\text{dist}(v_i, v_{i+1}) \leq \text{dist}(v_i, t)$. Assume that $\text{dist}(v_i, v_{i+1}) > \text{dist}(v_i, t)$, i.e., v_{i+1} is further away in the underlay graph from v_i than t. Since the prefix bit distance is improved in each step, it is $B_{v_i}(v_{i+1}) = B_{v_i}(t)$ which means that t would be stored in the same bucket of v_i as v_{i+1}. But if t is closer to v_i than v_{i+1} in the underlay graph and v_{i+1} is a contact of v_i, then vertex t must be a contact of v_i, too, since we assume an optimal overlay graph regarding Proximity Neighborhood Selection. In this case, v would directly route to t, and v_{i+1} would not be the next overlay hop. Thus, it is $\text{dist}(v_i, v_{i+1}) \leq \text{dist}(v_i, t) \leq \text{dist}(s,t) + \sum_{j=0}^{i-1} \text{dist}(v_j, v_{j+1})$ for all $i \in \{0, \ldots, h-1\}$. The second inequality is obtained by applying the triangle inequality.

We show that $\sum_{j=0}^{i-1} \text{dist}(v_j, v_{j+1}) \leq \left(2^i - 1\right) \cdot \text{dist}(s,t)$ for $i \in \{1, \ldots, h\}$. For $i = 1$, this is true by the previous observation. For $i > 1$, we have

$$\begin{aligned}
\sum_{j=0}^{i} \text{dist}(v_j, v_{j+1}) &= \sum_{j=0}^{i-1} \text{dist}(v_j, v_{j+1}) + \text{dist}(v_i, v_{i+1}) \\
&\leq \left(2^i - 1\right) \cdot \text{dist}(s,t) + \text{dist}(v_i, v_{i+1}) \\
&\leq \left(2^i - 1\right) \cdot \text{dist}(s,t) + \text{dist}(s,t) + \sum_{j=0}^{i-1} \text{dist}(v_j, v_{j+1}) \\
&\leq \left(2^i - 1\right) \cdot \text{dist}(s,t) + \text{dist}(s,t) + \left(2^i - 1\right) \cdot \text{dist}(s,t) = \left(2^{i+1} - 1\right) \cdot \text{dist}(s,t).
\end{aligned}$$

It follows directly that $2^{\text{dist}_p(s,t)} - 1$ is an upper bound for the stretch $S_{s,t}$. ∎

An example graph, where there are two vertices s and t with maximally possible stretch is depicted in Figure 4.1. Thus, the proposed bound is tight for a fixed prefix bit distance.

4.3 Random ID Assignment

In the previous section, we gave a worst-case upper bound disregarding that IDs are chosen randomly. In the following, we give an asymptotically matching lower bound that even holds in the probabilistic setting with random IDs. We basically build on the construction for the worst-case lower bound. However, using the construction from the previous section, the probability that KIRA routes in the wrong direction instead of routing directly to t decreases significantly with an increasing number of hops.

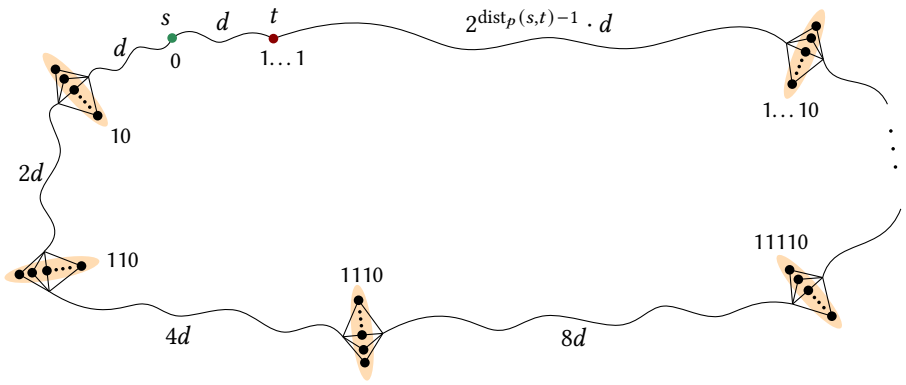

Figure 4.1: An example graph with an ID assignment such that $S_{s,t} = 2^{\text{dist}_P(s,t)} - 1$. Only the relevant prefixes of the IDs are specified, and the IDs of vertices on paths between *hubs* are 0-vertices. Each hub (marked in orange) consists of exactly k vertices, which share a prefix. The distance between s and t is d, and the length of the paths between hubs doubles after each hub. With this construction, it is possible that s routes over each hub before reaching t, correcting bits one by one, which gives us the proposed stretch.

In this section, we construct a family of graphs such that there is a pair of vertices for which the algorithm finds a path with high stretch with at least constant probability. After describing the topology of the graphs first, we qualify parameters that need to be chosen for the construction. Then, we analyze the probability distribution of the stretch that is achieved by the algorithm.

First, we introduce the notion of *interesting* vertices. Interesting vertices are vertices that are candidates for the next overlay hop, i.e., they make some prefix-wise improvement towards the target. More formally, a vertex $u \in V(G)$ is *v-t-interesting* for some $v \in V(G)$ and some target t if $\text{dist}_p(u,t) < \text{dist}_p(v,t)$. This is equivalent to $B_v(u) = B_v(t)$, i.e., the vertices u and t belong to the same bucket with depth $\text{lcp}(\text{ID}(v), \text{ID}(t))$. The following lemma shows that the probability that some vertex is v-t-interesting depends on the prefix bit distance between v and t. Note that this is independent of the underlay graph.

Lemma 4.2: *Let $v, t \in V(G)$ and $d := \text{dist}_p(v,t)$ the prefix bit distance between v and t. The probability that a vertex $u \in V(G)$ is v-t-interesting, is $\frac{2^{d-1}}{N}$.*

Proof. By definition, v-t-interesting vertices share a bucket of depth $\text{lcp}(\text{ID}(x), \text{ID}(t)) = B - d$ in the routing table of v, where B is the number of bits of each ID. Thus, these vertices share a prefix of length exactly $\text{lcp}(\text{ID}(v), \text{ID}(t))$ with v. The probability that a vertex has this property is

$$\left(\frac{1}{2}\right)^{\text{lcp}(\text{ID}(v),\text{ID}(t))+1} = \left(\frac{1}{2}\right)^{B-d+1} = \frac{2^{d-1}}{N}.$$

■

4.3.1 Construction of Graph

First, we describe the topology of the graph and introduce several parameters that need to be specified later. Let $h \in \mathbb{N}$. The graph consists of $h+1$ disjoint vertex sets $V_0, V_1, \ldots, V_h$, which we call *hubs*. Each hub forms an independent set, i.e., the subgraph induced by the vertices of a hub contains no edges. The hub V_0 only consists of one vertex s, which we call the *source*.

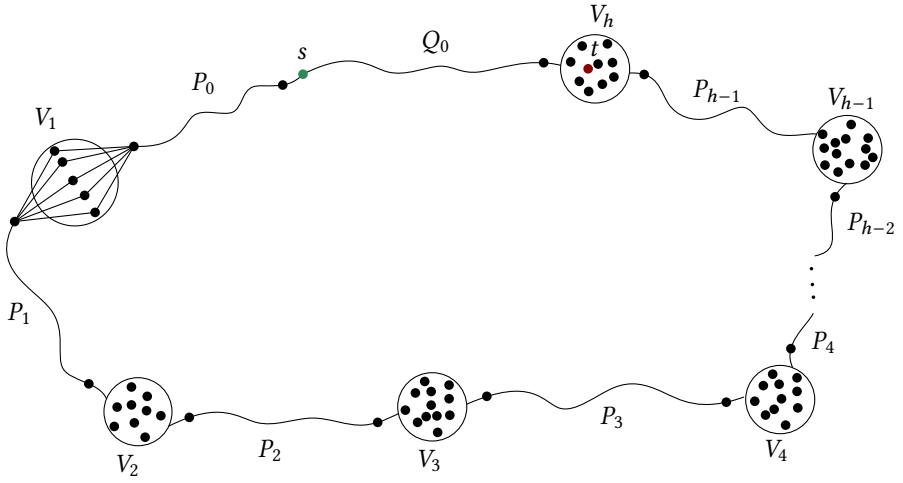

Figure 4.2: The constructed graph G with source s and target $t \in V_h$, h hubs $V_1, \dots V_h$, and the path Q_0 and the paths in $\mathcal{P}$. The graph is fully specified, given the number and sizes of the hubs and the lengths of the paths.

The hub V_h may contain more than one vertex, and there is a vertex in V_h that we call the *target*. As the names suggest, we are interested in the stretch of a path from s to some target in V_h found by the algorithm. All other hubs may contain any number of vertices. We denote the size of a hub V_i by n_i for $i \in \{1, \dots, h\}$.

The hubs $V_0, \dots, V_h$ are arranged on a cycle and are connected by pairwise disjoint paths in $\mathcal{P} = \{P_0, P_1, \dots, P_{h-1}\}$. For a path P_i with $i \in \{0, \dots, h-1\}$, one endpoint of P_i is adjacent to all vertices in V_i, whereas the other endpoint of P_i is adjacent to all vertices in V_{i+1}. Any inner vertex of P_i is not adjacent to any vertices other than its neighbors in P_i. Additionally, the vertices s and the vertices in V_h are connected by a path Q_0, i.e., all vertices in V_h are adjacent to one endpoint of Q_0, while s is adjacent to the other endpoint of Q_0. The structure of the graph is depicted in Figure 4.2

The idea is to force KIRA to route along a path from source s to some target in V_h such that the i-th overlay hop of the path is in hub V_i for all overlay hops on the path. Between two overlay hops, the algorithm takes a path in $\mathcal{P}$. We first specify the path lengths such that the resulting stretch depends on the number of overlay hops from s to the target. Further, we choose the size of each hub of the graph.

Path Lengths We specify the lengths of each path in $\mathcal{P}$ such that it is possible that KIRA finds a path from s to t as described above. Further, we want to maximize the stretch in case such a path found. As we are only interested in the distance between different hubs (which is not equal to the lengths of the paths in $\mathcal{P}$), we simplify the notation. Let $\text{dist}_{\mathcal{P}}(V_i, V_{i+1})$ with $i \in \{0, \dots, h-2\}$ be the distance between hub V_i and V_{i+1} if the path P_i is taken. Since every vertex in V_i has the same distance to any vertex in V_{i+1}, this is well defined. Note that it is $\text{dist}_{\mathcal{P}}(V_i, V_{i+1}) = |P_i| + 2$, thus, it is sufficient to specify each $\text{dist}_{\mathcal{P}}(V_i, V_{i+1})$ instead of the lengths of each path in $\mathcal{P}$. We further introduce a new variable ℓ and variables ℓ_i for $i \in \{0, \dots, h-1\}$, where ℓ_i is the distance $\text{dist}_{\mathcal{P}}(V_i, V_{i+1})$. Each ℓ_i depends on ℓ, which roughly describes the distance between s and V_h.

We choose the lengths in such a way that they satisfy some convenient properties under the assumption that the i-th overlay hop is indeed in the i-th hub V_i: On the one hand, our goal is to maximize the stretch of the found path, thus we choose the paths as long as possible. However, on the other hand, we have to make sure that the vertices in V_{i+1} are closer to the vertices in V_i than the target. Otherwise, a vertex in V_i would prefer to store the target. In this case, the algorithm would directly route to t from the i-th overlay hop. With these two constraints in mind, we set $\ell_i = 2^i \cdot \ell$.

If the path goes through all hubs $V_0, \dots, V_h$, we obtain maximal stretch, similar as in Lemma 4.1. However, it is possible that the found path goes over an overlay hop $v \in V(G)$ that knows the target directly as a contact. In this case, the algorithm takes the path from v to the target, which is stored in the routing table of v. Recall that this underlay path is a shortest path from v to t since we assume an optimal overlay graph regarding Proximity Neighborhood Selection. The way we chose the lengths of the paths in $\mathcal{P}$, such a shortest path from a vertex v in some hub V_i with $i \in \{1, \dots, h-2\}$ to a target in V_h always routes back to s, going over the paths $P_{i-1}, \dots, P_0$. From s, the path routes to a target in V_h via the path Q_0. Since the algorithm optimizes the found path by erasing all loops, the resulting path would be optimal in that case. Although it does not affect the worst case, we still aim to achieve some stretch even if the found path does not go via all hubs $V_1, \dots, V_{h-1}$. Thus, we introduce new paths $Q_1, \dots, Q_{h-2}$ that connect each hub with the vertices in V_h by a path that is slightly shorter than the path via s and thus preferred. One endpoint of Q_i with $i \in \{1, \dots, h-2\}$ is adjacent to all vertices in V_i and the other endpoint is adjacent to all vertices in V_h. Let $\mathcal{Q} = \{Q_0, \dots, Q_{h-2}\}$, where Q_0 is the path that connects source s with the vertices in V_h.

We choose the lengths of each path in $\mathcal{Q}$ such that at a hub V_i, a contact in the next hub V_{i+1} is preferred over the target for $i \in \{1, \dots, h-2\}$. In other words, Q_i needs to be longer than P_i for every $i \in \{0, \dots, h-2\}$. Similarly as before, we denote the distance between a hub V_i and the target hub V_h that goes over Q_i by $\text{dist}_{\mathcal{Q}}(V_i, V_h)$ for $i \in \{0, \dots, h-2\}$. We set $\text{dist}_{\mathcal{Q}}(V_i, V_h) := \ell_i + 1$ for $i \in \{1, \dots, h-2\}$ and $\text{dist}_{\mathcal{Q}}(V_0, V_h) := \ell_0 + 2$. Figure 4.3 shows a sketch of graph G with the specified lengths of the paths in $\mathcal{P}$ and $\mathcal{Q}$.

The following lemma shows that the stretch of a found path grows exponentially with the number of hubs it routes through. Note that this asymptotically matches the upper bound given in Lemma 4.1 for worst-case ID assignments.

Lemma 4.3: *Let $R = (r_0 = s, r_1, \dots, r_{j-1}, t)$ with $r_i \in V_i$ for $i \in \{1, \dots, j-1\}$ and $t \in V_h$ be a path in G from s to a target t in V_h. It is possible that KIRA finds this path when t is requested by s. For $\ell \geq 2$, the stretch of the found path is at least 2^{j-1}.*

Proof. We first prove that it is possible for the algorithm to find such a route. Recall that the shortest path from hub V_i to hub V_{i+1} for all $i \in \{0, \dots, h-1\}$ has length $\text{dist}_{\mathcal{P}}(V_i, V_{i+1}) = \ell_i = 2^i \cdot \ell$. Further, the distance between s and t is $\ell_0 + 2$. We first determine the distance between an overlay hop $r_i \in R$ and $t \in V_h$ by comparing the lengths of different paths from r_i to t. The length of the path between r_i and t using path Q_i is $\text{dist}_{\mathcal{Q}}(V_i, V_h) = \ell_i + 1$ by definition. Another possible path goes back to s, taking the overlay hops $r_{i-1}, \dots, r_1, s$ in reversed order. From there, it routes to t via Q_0. The length of this path is

$$\sum_{j=0}^{i-1} \text{dist}_{\mathcal{P}}(V_j, V_{j+1}) + \text{dist}_{\mathcal{Q}}(s, t) = \sum_{j=0}^{i-1} \ell_j + \ell_0 + 2 = \sum_{j=0}^{i-1} 2^j \cdot \ell + \ell_0 + 2 = \left(2^i - 1\right) \cdot \ell + \ell_0 + 2 = \ell_i + 2.$$

All other paths are clearly longer. Thus, the distance between $r_i \in V_i$ and the target is $\text{dist}(r_i, t) = \text{dist}_{\mathcal{Q}}(V_i, V_h)$. As KIRA prefers shorter paths, it takes a path in $\mathcal{Q}$ rather than routing back to s and from there to t if the current overlay hop knows t as a target.

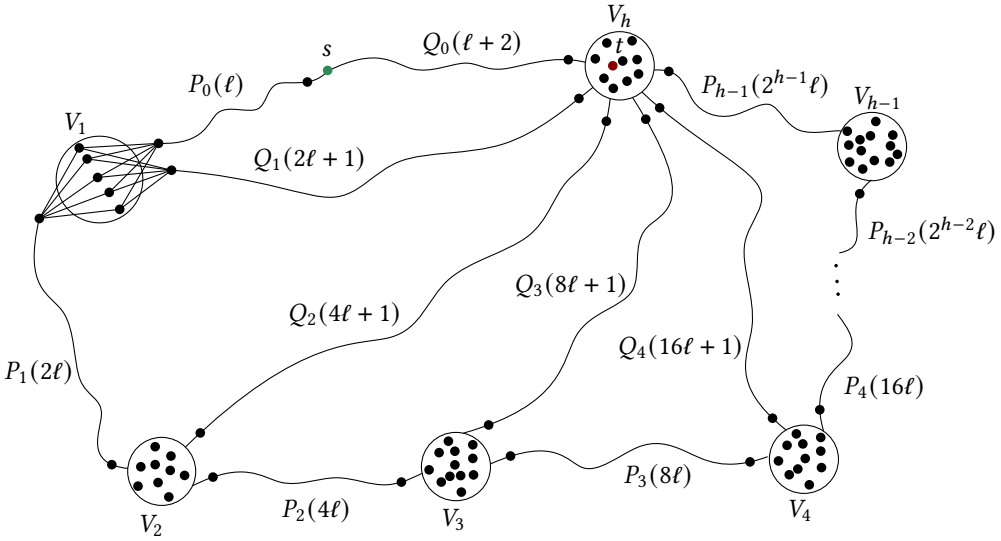

Figure 4.3: The constructed graph G with source s and target $t \in V_h$, h hubs $V_1, \dots V_h$, and paths P_i for $i \in \{0, \dots, h-1\}$ and Q_i for $i \in \{0, \dots, h-2\}$ which connect pairs of hubs. For each path that connects two hubs V' and V'', we specify $\text{dist}_{\mathcal{P}}(V', V'')$ and $\text{dist}_{\mathcal{Q}}(V', V'')$, respectively, in brackets after the name of the path.

Further, we need to prove that a vertex in V_i prefers to store a vertex in V_{i+1} in its routing table over t for $i \in \{0, \dots, h-2\}$. The distance between a vertex in V_i and a vertex in V_{i+1} is $\text{dist}(V_i, V_{i+1}) = \text{dist}_{\mathcal{P}}(V_i, V_{i+1}) = \ell_i < \ell_i + 1 = \text{dist}(V_i, t)$ for $i \in \{1, \dots, h-2\}$. For $i = 0$, we have $\text{dist}(V_0, V_1) = \ell_0 < \ell_i + 2 = \text{dist}(V_0, t)$. As the underlay distance to the next hub V_{i+1} is smaller than t, the vertices in V_{i+1} are preferred as contacts over the target. Thus, it is possible that no overlay hop in R (except for the last hop before t) knows the target as a contact.

For the length of R, we obtain the following sum if we assume $\ell \geq 2$.

$$|R| = \sum_{i=0}^{j-1} \text{dist}(r_i, r_{i+1}) \;\geq \sum_{i=0}^{j-1} \ell_i \;= \ell \cdot \sum_{i=0}^{j-1} 2^i \;= \ell \cdot \left(2^j - 1\right) \;\geq (\ell+2) \cdot 2^{j-1}.$$

This gives us a stretch of at least 2^{j-1}. ∎

Hub Sizes To make it probable that the path goes via the hubs $V_1, \dots, V_{h-1}$, the main idea is to make each hub sufficiently large such that the probability that the algorithm routes from a vertex in V_i to a vertex in V_{i+1} is high for some given target. Assume that $v_i \in V_i$ is the i-th overlay hop and t is the target. If $B_{v_i}(t) \subseteq V_{i+1}$, then all candidates for the next overlay hop are in V_{i+1}. For this to happen with sufficiently high probability, we need sufficiently many vertices in V_{i+1} that are v_i-t-interesting. Thus, we aim to choose n_i, the size of hub V_i, large enough such this condition is satisfied.

From now on, we refer to this desired number of interesting vertices in each hub as the constant r. Lemma 4.2 implies that the number of interesting vertices also depends on the prefix bit distance of the current overlay hop to the target. Thus, the choice of the size of V_{i+1} needs to take the bit distance of the i-th overlay hop to the target into account. More specifically, as the bit distance to the target decreases at each overlay hop, the size of each hub has to increase along the path to maintain a constant number of r interesting vertices in each hub.

Let d_0^* denote the desired initial prefix bit distance between s and some target in V_h. For fixed vertices s and t, it is not very likely that s and t actually have bit distance d_0^*. We boost this probability by choosing the size of the last hub V_h, which contains the target, sufficiently large. We set the size of the last hop $|V_h| = n_h := \frac{N}{2^{d_0^*-1}}$. The following lemma shows that the probability that s and a vertex in V_h have the desired initial prefix bit distance is constant.

Lemma 4.4: *Let d_0^* be the desired prefix bit distance between the source and the target. The probability that s and a vertex in V_h have prefix bit distance d_0^* is at least $1 - \frac{1}{e}$.*

Proof. First, consider a fixed vertex $v \in V_h$. It has prefix bit distance d_0^* to s if and only if $\mathrm{lcp}(\mathrm{ID}(s), \mathrm{ID}(v)) = B - d_0^*$. The probability that this happens is

$$\left(\frac{1}{2}\right)^{\mathrm{lcp}(\mathrm{ID}(s),\mathrm{ID}(v))+1} = \left(\frac{1}{2}\right)^{B-d_0^*+1} = \frac{2^{d_0^*-1}}{N}.$$

Then, the probability that there is a vertex in V_h with the desired prefix bit distance is

$$1 - \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{n_h} = 1 - \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{\frac{N}{2^{d_0^*-1}}} \geq 1 - \frac{1}{e}.$$

■

In the following, we assume that there is a vertex $t \in V_h$ with bit distance d_0^*, and we consider the request from source s to target t.

The improvement of the bit distance towards the target may vary after each overlay hop, but similarly as for r, the number of desired interesting vertices after each hop, we fix a desired amount of improvement. At each overlay hop, we aim to improve the bit distance to the target by b^* bits. For now, we define $b^* := \log\left(\frac{r}{\ln(2)}\right)$. We later show b^* is indeed appropriately chosen and that the improvement at each overlay hop is not too far from b^* with sufficiently high probability. With the definition of b^*, we can now define the corresponding desired bit distances $d_i^* := d_0^* - i \cdot b^*$ between the i-th overlay hop on the found path and the target. Depending on the desired bit distances, we set the size n_i of a hub V_i with $i \in \{1, \ldots, h-1\}$ to

$$n_i := \frac{N \cdot r}{2^{d_{i-1}^*-1}} = \frac{N \cdot \ln(2) \cdot 2^{b^*} \cdot 2^{(i-1)\cdot b^*}}{2^{d_0^*-1}} = \ln(2) \cdot N \cdot 2^{i \cdot b^* - d_0^*+1}. \tag{4.1}$$

Note that by choosing values for ℓ, r, n_h, d_0^* and h, the graph is fully specified.

4.3.2 Lower Bound for the Probability

In the following, we determine a lower bound for the probability that the graph satisfies our conditions, i.e., that the path found by the algorithm from s to t goes over all hubs. We first consider each pair of consecutive overlay hops separately and combine these probabilities to obtain a result for the whole path.

Let Y_i be a random variable that is 1 if the i-th overlay hop on the chosen route is in V_i and 0 otherwise for $i \in \{1, \ldots, h-1\}$. We consider the probability of going from some hub V_i to the next hub V_{i+1} in one step, i.e., the probability $\Pr[Y_{i+1} = 1 \mid Y_i = 1]$. Apart from the number of vertices in V_{i+1}, this probability also depends on the prefix bit distance between the i-th overlay hop and the target, which we call D_i.

For the construction of the graph, we assumed the desired prefix bit distance $d_0^* - i \cdot b^*$ after the i-th overlay hop. However, we need to consider that the bit improvement at each overlay hop may vary. If the i-th overlay hop for some $i \in \{1, \dots, h-1\}$ improves the bit distance much more than the desired improvement b^*, then it is possible that there are not sufficiently many interesting vertices in V_{i+1}. On the other hand, if it improves the bit distance too little, then it is likely that V_{i+1} contains too many interesting vertices and the algorithm might overcompensate by choosing a vertex as the next hop that improves the bit distance far too much. To measure this deviation on the improvement at each overlay hop, we introduce a new random variable $\Delta_i \coloneqq D_i - d_i^*$ for every $i \in \{0, \dots, h-1\}$, where d_i^* refers to the desired prefix bit distance to the target after the i-th overlay hop. If Δ_i is negative, the algorithm has improved the distance more than desired. The following lemma determines $\mathbb{E}[X_{i+1} \mid \Delta_i]$, the expected value of X_{i+1}, depending on the deviation Δ_i after the i-th overlay hop.

Lemma 4.5: *For the i-th overlay hop with $i \in \{0, \dots, h-2\}$, it is $\mathbb{E}[X_{i+1} \mid \Delta_i] = r \cdot 2^{\Delta_i}$.*

Proof. Let $i \in \{0, \dots, h-2\}$ and v_i be the i-th overlay hop of the found path. By assumption, the prefix bit distance between v_i and t is $D_i = d_0^* - i \cdot b^* + \Delta_i$. Let X_v for $v \in V$ be an indicator random variable that is 1 if v is v_i-t-interesting and 0 if not. Then, by Lemma 4.2, we have

$$\Pr[X_v = 1 \mid \Delta_i] = \frac{2^{d_0^* - i \cdot b^* + \Delta_i - 1}}{N}$$

for every $v \in V$. For the expected value of v_i-t-interesting vertices in V_{i+1}, we obtain

$$\begin{aligned}\mathbb{E}[X_i \mid \Delta_i] &= \mathbb{E}\left[\sum_{v \in V_{i+1}} X_v \,\middle|\, \Delta_i\right] = \sum_{v \in V_{i+1}} \mathbb{E}[X_v \mid \Delta_i] = n_{i+1} \cdot \frac{2^{d_0^* - i \cdot b^* + \Delta_i - 1}}{N} \\ &= \ln(2) \cdot N \cdot 2^{i \cdot b^* - d_0^* + 1} \cdot \frac{2^{d_0^* - i \cdot b^* + \Delta_i - 1}}{N} = \ln(2) \cdot 2^{b^* + \Delta_i} = r \cdot 2^{\Delta_i}\end{aligned}$$

by using $n_{i+1} = \ln(2) \cdot N \cdot 2^{i \cdot b^* - d_0^* + 1}$ as defined in Equation 4.1. ∎

As $\mathbb{E}[X_{i+1} \mid \Delta_i = \delta]$ does not depend on i, we ignore the index and refer to it as $\mu(\delta)$. Further, note that if the deviation of the prefix bit distance after the i-th overlay hop is 0, then we have $\mu(0) = r$, which is exactly the desired number of interesting vertices.

To simplify, we are only interested in the case where deviations after each overlay hop are not too negative, i.e., the actual bit distance to the target does not deviate too much from the desired bit distance. Later, we show that this assumption is reasonable by proving that it holds with sufficiently high probability after each hop. We refer to the smallest deviation we still consider as $\delta_L < 0$, which we will choose only depending on N, the size of the ID space. More formally, for all $i \in \{0, \dots, h-2\}$, we determine the probability that the i-th overlay hop of the found path is in V_i under the condition that $\Delta_i \geq \delta_L$ is satisfied.

The following lemma gives a lower bound for $\Pr[Y_{i+1} = 1 \mid \Delta_i \geq \delta_L, Y_i = 1]$, the probability that the $(i+1)$-th overlay hop is in V_{i+1} if the i-th overlay hop is in V_i and if the deviation is not too negative. We denote this probability by $P_Y(i+1, \delta_L)$.

Lemma 4.6: *The probability that the $(i+1)$-th overlay hop is a vertex in V_{i+1} if the i-th overlay hop is a vertex in V_i, given the deviation Δ_i after the i-th hop is*

$$\Pr[Y_{i+1} = 1 \mid \Delta_i, Y_i = 1] \geq \left(1 - \frac{2^{d_0^* - 1}}{N}\right)^{|V(P_i)| + |V(Q_i)|} \cdot \left(1 - \exp\left(\frac{-\mu(\Delta_i)}{2} + k\right)\right).$$

Further, if we condition on the event that $\Delta_i \geq \delta_L$, we obtain the following lower bound for the probability:

$$\Pr\left[Y_{i+1} = 1 \mid \Delta_i \geq \delta_L, Y_i = 1\right] \geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{|V(P_i)|+|V(Q_i)|} \cdot \left(1 - \exp\left(\frac{-\mu(\delta_L)}{2} + k\right)\right).$$

Proof. We assume that the i-th overlay hop is a vertex $v_i \in V_i$ with prefix bit distance $D_i = d_i^* + \Delta_i = d_0^* - i \cdot b^* + \Delta_i$ to t. Then, the $(i+1)$-th overlay is in V_{i+1} if $B_{v_i}(t) \subseteq V_{i+1}$. In this case, all v_i-t-interesting vertices that v_i knows as contacts are in V_{i+1}, which forces the next overlay hop to be in V_{i+1}. Since we assume that the i-th overlay hop is in V_i, vertices in the previous hubs $V_0, \ldots, V_i$ and on the paths $P_0, \ldots, P_{i-1}$ and $Q_0, \ldots, Q_{i-1}$ cannot be interesting at the i-th overlay hop. Thus, we only need to consider vertices in P_i, Q_i and V_{i+1}. The following two conditions are sufficient to ensure $B_{v_i}(t) \subseteq V_{i+1}$:

1. No vertex in P_i or Q_i is v_i-t-interesting. Otherwise, such a vertex would be closer in the underlay graph to v_i than any vertex in V_{i+1}, and it would be preferred as a contact in $B_{v_i}(t)$ over any vertex in V_{i+1}. Let A_i be the event that no vertex on $P_i \cup Q_i$ is v_i-t-interesting after the i-th overlay hop.

2. There are at least k vertices in V_{i+1} that are v_i-t-interesting, i.e., $X_{i+1} \geq k$. Since the vertices in V_{i+1} are closer to v_i than t by construction, vertices in V_{i+1} are preferred as contacts over t.

These two conditions are independent of each other. We first determine the probability that the above conditions are satisfied separately for each condition.

By Lemma 4.2, the probability that a vertex $v \in V(G)$ is v_i-t-interesting is $\frac{2^{d_0^*-i\cdot b^*+\Delta_i-1}}{N}$. Further, we know that the prefix bit distance $D_i = d_0^* - i \cdot b^* + \Delta_i$ between v_i and t is at most d_0^* since the prefix bit distance decreases at each overlay hop. Thus, the probability that no vertex in Q_i and P_i is v_i-t-interesting, is

$$\begin{aligned}\Pr\left[A_i \mid \Delta_i\right] &= \left(1 - \frac{2^{d_0^*-i\cdot b^*+\Delta_i-1}}{N}\right)^{|V(P_i)|+|V(Q_i)|} \\ &\geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{|V(P_i)|+|V(Q_i)|}.\end{aligned}$$

For the second part, we obtain a lower bound for $\Pr\left[X_{i+1} \geq k \mid \Delta_i\right]$. We first rearrange the equation such that Chernoff's bound is applicable.

$$\begin{aligned}\Pr\left[X_{i+1} \geq k \mid \Delta_i\right] &\geq 1 - \Pr\left[X_{i+1} \leq k \mid \Delta_i\right] \\ &= 1 - \Pr\left[X_{i+1} \leq 1 - \left(1 - \frac{k}{\mu(\Delta_i)}\right) \cdot \mu(\Delta_i) \;\middle|\; \Delta_i\right]\end{aligned}$$

Recall that the expected value $\mathbb{E}\left[X_{i+1} \mid \Delta_i\right]$ is $\mu(\Delta_i)$. Further, note that X_{i+1} can be written as the sum of the 0-1-valued random variables X_v for $v \in V_{i+1}$, where X_v is 1 if v is v_i-t-interesting and 0 if not. By assumption, the random variables are independently and identically distributed. Thus, we can apply the first Chernoff bound (2.3) with $\alpha := 1 - \frac{k}{\mu(\Delta_i)}$ to obtain the following lower bound:

$$\begin{aligned}&\geq 1 - \exp\left(-\left(1 - \frac{k}{\mu(\Delta_i)}\right)^2 \cdot \frac{\mu(\Delta_i)}{2}\right) = 1 - \exp\left(-\frac{\mu(\Delta_i)}{2} + k - \frac{k^2}{2 \cdot \mu(\Delta_i)}\right) \\ &\geq 1 - \exp\left(-\frac{\mu(\Delta_i)}{2} + k\right).\end{aligned}$$

Now, we combine the probabilities for each condition to determine a lower bound for $\Pr[Y_{i+1} = 1 \mid Y_i = 1, \Delta_i]$:

$$\begin{aligned}\Pr[Y_{i+1} = 1 \mid \Delta_i, Y_i = 1] &\geq \Pr[A_i, X_{i+1} \geq k \mid \Delta_i] = \Pr[A_i \mid \Delta_i] \cdot \Pr[X_{i+1} \geq k \mid \Delta_i] \\ &\geq \left(\left(1 - \frac{2^{d_0^*-1}}{n}\right)^{|V(P_i)|+|V(Q_i)|}\right) \cdot \left(1 - \exp\left(-\frac{\mu(\Delta_i)}{2} + k\right)\right).\end{aligned}$$

This proves the first inequality. To prove the second inequality, we rewrite the probability for a fixed lower bound δ_L as follows.

$$\begin{aligned}&\Pr[Y_{i+1} = 1 \mid \Delta_i \geq \delta_L, Y_i = 1] \\ &= \sum_{\delta} \Pr[Y_{i+1} = 1 \mid \Delta_i = \delta, \Delta_i \geq \delta_L, Y_i = 1] \cdot \Pr[\Delta_i = \delta \mid \Delta_i \geq \delta_L, Y_i = 1]\end{aligned}$$

As $1 - \exp\left(-\frac{\mu(\delta)}{2} + k\right)$ decreases for decreasing δ, we obtain a lower bound of the first factor with $\Delta_i = \delta_L$:

$$\begin{aligned}&\geq \left(1 - \exp\left(-\frac{\mu(\delta_L)}{2} + k\right)\right) \cdot \left(\left(1 - \frac{2^{d_0^*-1}}{N}\right)^{|V(P_i)|+|V(Q_i)|}\right) \cdot \sum_{\delta} \Pr[\Delta_i = \delta \mid \Delta_i \geq \delta_L, Y_i = 1] \\ &= \left(1 - \exp\left(-\frac{\mu(\delta_L)}{2} + k\right)\right) \cdot \left(\left(1 - \frac{2^{d_0^*-1}}{N}\right)^{|V(P_i)|+|V(Q_i)|}\right) \cdot 1.\end{aligned}$$

■

In the following, we determine the probability distribution of the deviation Δ_{i+1} after the $(i+1)$-th overlay hop, given that the i-th hop is in V_i and that the $(i+1)$-th hop is in V_{i+1} and given $\Delta_i \geq \delta_L$. Intuitively speaking, we have the following situation: The i-th overlay hop is some vertex v in V_i, and we know that the next overlay hop is one of the v-t-interesting vertices in V_{i+1}. Out of those, the algorithm chooses the vertex that is bitwise closest to t. We are now interested in the deviation Δ_{i+1} after choosing the $(i+1)$-th hop if $\Delta_i \geq \delta_L$. To calculate $\Pr[\Delta_{i+1} \geq \delta_L \mid \Delta_i \geq \delta_L, Y_i = 1, Y_{i+1} = 1]$, we first determine how likely it is to improve a certain amount of bits at an overlay hop in the following lemma.

Lemma 4.7: *Let v be a vertex in $V(G)$, t the target and $\mathrm{dist}_p(v,t)$ the prefix bit distance between v and t. Let X be the set of v-t-interesting candidates for the next overlay hop. Further, let $b \in \mathbb{N}$. Then, the probability that the amount of improvement $\mathrm{dist}_p(v,t) - \mathrm{dist}_p(u,t)$ is at most b for every $u \in X$ is*

$$\left(1 - 2^{-b}\right)^{|X|}.$$

Proof. Let $u \in X$ be a v-t-interesting candidate for the next overlay hop. Then, we know that $\mathrm{dist}_p(v,t) - \mathrm{dist}_p(u,t) \geq 1$, i.e., u improves the prefix bit distance to t by at least 1. For u to improve the prefix bit distance by exactly b bits for some $b \in \{1, \ldots, \mathrm{dist}_p(v,t)\}$, the ID of u needs to share an additional $b-1$ bits with the ID of t, and the bit after needs to be different. The probability that the prefix bit distance to t improves by exactly b bits is 2^{-b}. Then, the probability that u improves the prefix bit distance by at most b bits is

$$\sum_{i=1}^{b} 2^{-i} = \frac{2^{-b-1} - 1}{2^{-1} - 1} = 1 - 2^{-b}.$$

Since we assume that the IDs are independently assigned, the probability that all vertices in X improve the prefix bit distance by at most b is $\left(1 - 2^{-b}\right)^{|X|}$. ∎

Lemma 4.8: *Let $\alpha \geq 0$. Then, it is*

$$\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i, Y_i = 1, Y_{i+1} = 1\right] \geq 2^{-2^{\delta_L+1}\cdot(1+\alpha)} \cdot \left(1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\Delta_i)}{2+\alpha}\right)\right).$$

Further, we obtain a lower bound for this probability if Δ_i is at least δ_L:

$$\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i \geq \delta_L, Y_i = 1, Y_{i+1} = 1\right] \geq 2^{-2^{\delta_L+1}\cdot(1+\alpha)} \cdot \left(1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\delta_L)}{2+\alpha}\right)\right).$$

Proof. The probability distribution Δ_{i+1} depends on the probability distribution of X_{i+1} given the deviation Δ_i. If X_{i+1} is too large, then the improvement of bits is likely to be too large as well. On the other hand, it is unlikely that X_{i+1} is much smaller than the expected number of interesting vertices. We obtain a lower bound by only considering cases where X_{i+1} is not too much larger than the expected value (specifically, $X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)$), and we show that $\Delta_{i+1} \geq \delta_L$ holds with sufficiently high probability, even with this additional restriction. Since we now assume that the i-th overlay hop is in V_i, and the $(i+1)$-th overlay hop is in V_{i+1} (i.e., $Y_i = Y_{i+1} = 1$), we omit both conditions in the following for better readability. Then, we estimate a lower bound as follows.

$$\begin{aligned}
&\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i\right] \geq \Pr\left[\Delta_{i+1} \geq \delta_L, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i) \mid \Delta_i\right] \\
&\geq \Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right] \cdot \Pr\left[X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i) \mid \Delta_i\right]
\end{aligned}$$

We first consider both factors separately. Again, X_{i+1} is the sum of 0-1-valued random variables that are independently and identically distributed. Thus, we can apply the second Chernoff bound (2.4) to the second factor:

$$\Pr\left[X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i) \mid \Delta_i\right] \geq 1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\Delta_i)}{2+\alpha}\right)$$

For the first factor, we get:

$$\begin{aligned}
&\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right] \\
&\geq \sum_{x}^{(1+\alpha)\cdot\mu(\Delta_i)} \Pr\left[\Delta_{i+1} \geq \delta_L, X_{i+1} = x \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right] \\
&\geq \sum_{x}^{(1+\alpha)\cdot\mu(\Delta_i)} \Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i, X_{i+1} = x\right] \cdot \Pr\left[X_{i+1} = x \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right]
\end{aligned}$$

The first factor represents the probability that the prefix bit distance to the target improves by at most $D_i - D_{i+1} \leq d_0^* + i\cdot b^* - (d_0^* + (i+1)\cdot b^* + \delta_L) = b^* + \Delta_i - \delta_L$ bits if there are x interesting candidates for the next overlay hop. By Lemma 4.7, this probability is equal to $\left(1 - 2^{-(b^*+\Delta_i-\delta_L)}\right)^x$.

$$= \sum_{x}^{(1+\alpha)\cdot\mu(\Delta_i)} \left(1 - 2^{-(b^*+\Delta_i-\delta_L)}\right)^x \cdot \Pr\left[X_{i+1} = x \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right]$$

As the first factor decreases with increasing x, we obtain a lower bound for $x = (1+\alpha) \cdot \mu(\Delta_i)$:

$$\geq \left(1 - 2^{-(b^*+\Delta_i-\delta_L)}\right)^{(1+\alpha)\cdot\mu(\Delta_i)} \cdot \sum_{x}^{(1+\alpha)\cdot\mu(\Delta_i)} \Pr\left[X_{i+1} = x \mid \Delta_i, X_{i+1} \leq (1+\alpha)\cdot\mu(\Delta_i)\right]$$
$$= \left(1 - 2^{-(b^*+\Delta_i-\delta_L)}\right)^{(1+\alpha)\cdot\mu(\Delta_i)} \cdot 1$$

By definition of b^*, we have $\mu(\Delta_i) = \ln(2) \cdot 2^{b^*+\Delta_i}$. Now, we can rewrite the last line and use the inequality $\left(1 - x^{-1}\right)^x \geq \frac{1}{2e}$:

$$= \left(1 - 2^{-(b^*+\Delta_i-\delta_L)}\right)^{2^{b^*+\Delta_i-\delta_L}\cdot\ln(2)\cdot 2^{\delta_L}\cdot(1+\alpha)} \geq (2e)^{-\ln(2)\cdot 2^{\delta_L}\cdot(1+\alpha)} \geq (e^2)^{-\ln(2)\cdot 2^{\delta_L}\cdot(1+\alpha)}$$
$$= 2^{-2^{\delta_L+1}\cdot(1+\alpha)}.$$

Note that this lower bound does not depend on the deviation Δ_i after the i-th overlay hop. Multiplying the lower bounds for each factor gives us the proposed lower bound for

$$\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i, Y_i = 1, Y_{i+1} = 1\right].$$

For the second inequality, we rewrite the probability for a fixed lower bound δ_L as follows.

$$\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i \geq \delta_L\right]$$
$$= \sum_{\delta} \Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i = \delta, \Delta_i \geq \delta_L\right] \cdot \Pr\left[\Delta_i = \delta \mid \Delta_i \geq \delta_L\right]$$

As $\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i = \delta\right]$ decreases with decreasing δ, we obtain a lower bound for the first factor with $\Delta_i = \delta_L$:

$$\geq 2^{-2^{\delta_L}\cdot(1+\alpha)} \cdot \left(1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\delta_L)}{2+\alpha}\right)\right) \cdot \sum_{\delta} \Pr\left[\Delta_i = \delta \mid \Delta_i \geq \delta_L\right]$$
$$= 2^{-2^{\delta_L}\cdot(1+\alpha)} \cdot \left(1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\delta_L)}{2+\alpha}\right)\right) \cdot 1.$$

∎

Depending on $\alpha \geq 0$, we denote the probability $\Pr\left[\Delta_{i+1} \geq \delta_L \mid \Delta_i \geq \delta_L, Y_i = 1, Y_{i+1} = 1\right]$ by $P_\Delta(\alpha, \delta_L)$. Now, we summarize the results of Lemma 4.6 and Lemma 4.8 to obtain a lower bound for the probability that the found path between s and t passes through a vertex in V_{i+1}.

Lemma 4.9: *For fixed $\delta_L < 0$, the probability that the i-th overlay hop is in V_i is*

$$Pr[Y_{i+1} = 1] \geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{2^{i+2}\cdot\ell} \cdot \left(2^{-2^{\delta_L+2}}\right)^{i+1} \cdot \left(1 - \exp\left(-\frac{\mu(\delta_L)}{3}\right)\right)^{2(i+1)}.$$

Proof. We restrict the set of possible events further by having the additional condition that $\Delta_{i+1} \geq \delta_L$, and show that the probability is still sufficiently high. We determine the probability by using the probability that the i-th overlay hop is in V_i with $\Delta_i \geq \delta_L$ and then, making an additional hop from V_i to V_{i+1}, which gives us

$$\Pr\left[Y_{i+1} = 1\right] \geq \Pr\left[Y_{i+1} = 1, \Delta_{i+1} \geq \delta_L\right] \geq P_Y(i+1, \delta_L) \cdot P_\Delta(\alpha, \delta_L) \cdot \Pr\left[Y_i = 1, \Delta_i \geq \delta_L\right].$$

Now, we use recursion to determine the probability explicitly. Applying Lemma 4.6 and Lemma 4.8 yields:

$$
\begin{aligned}
&= \prod_{j=0}^{i+1} P_Y(j, \delta_L) \cdot P_\Delta(\alpha, \delta_L)^{i+1} \cdot \Pr\left[Y_0 = 1, \Delta_0 \geq \delta_L\right] \\
&= \prod_{j=0}^{i+1} \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{|V(P_j)|+|V(Q_j)|} \cdot \left(1 - \exp\left(\frac{-\mu(\delta_L)}{2} + k\right)\right)^{i+1} P_\Delta(\alpha, \delta_L)^{i+1} \cdot 1 \\
&\geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{2^{i+2}\cdot\ell} \cdot \left(1 - \exp\left(\frac{-\mu(\delta_L)}{2} + k\right)\right)^{i+1} \cdot \left(2^{-2^{\delta_L+1}\cdot(1+\alpha)}\right)^{i+1} \cdot \left(1 - \exp\left(-\alpha^2 \cdot \frac{\mu(\delta_L)}{2+\alpha}\right)\right)^{i+1}
\end{aligned}
$$

With $\mu(\delta_L) \gg k$ and choosing $\alpha = 1$, we obtain a simplified lower bound:

$$
\geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{2^{i+2}\cdot\ell} \cdot \left(2^{-2^{\delta_L+2}}\right)^{i+1} \cdot \left(1 - \exp\left(-\frac{\mu(\delta_L)}{3}\right)\right)^{2(i+1)}.
$$

■

We have now proven a lower bound, which depends on d_0^*, ℓ and δ_L, for the probability that the algorithm routes over a hub V_i for some $i \in \{1, \ldots, h-1\}$. In the following, we aim to choose the parameters such that the probability for $\Pr\left[Y_{i+1} = 1\right]$ is sufficiently high. For a fixed number of hops h, we are interested in determining α, δ_L, r and δ_0^* such that $\Pr\left[Y_{h-1} = 1\right]$ is constant.

Theorem 4.10: *Let $h \in \mathbb{N}$ be the number of hops. Further, let $\delta_L \leq \log\left(-\log\left(1 - \frac{1}{h}\right)\right) - 2$, $\ell \leq N \cdot 2^{-d_0^*-h+1}$, and $r \geq -12 \cdot \ln(h)/\log(1 - \frac{1}{h})$. The probability that the $(h-1)$-th hop is in V_{h-1} is*

$$\Pr\left[Y_{h-1} = 1\right] \geq (2e)^{-4}.$$

Proof. We obtain the following constant lower bounds for each factor of the probability $\Pr\left[Y_{h-1} = 1\right]$ from Lemma 4.9. With $\ell \leq N \cdot 2^{-d_0^*-h+1}$, we can rewrite the first inequality as follows:

$$\left(1 - \frac{2^{d_0^*-1}}{N}\right)^{2^h\cdot\ell} \geq \left(1 - \frac{2^{d_0^*-1}}{N}\right)^{N\cdot 2^{-d_0^*+1}} \geq (2e)^{-1}.$$

For the second inequality, we have with $\delta_L \leq \log\left(-\log\left(1 - \frac{1}{h}\right)\right) - 2$:

$$\left(2^{-2^{\delta_L+2}}\right)^{h-1} \geq \left(2^{-2^{\log\left(-\log\left(1-\frac{1}{h}\right)\right)-2+2}}\right)^{h-1} = \left(1 - \frac{1}{h}\right)^{h-1} \geq (2e)^{-1}.$$

Further, it is

$$
\begin{aligned}
\left(1 - \exp\left(-\frac{\mu(\delta_L)}{3}\right)\right)^{2h-2} &= \left(1 - \exp\left(-\frac{r \cdot 2^{\delta_L}}{3}\right)\right)^{2h-2} \\
&\geq \left(1 - \exp\left(-\frac{1}{3} \cdot 12 \cdot \frac{-\ln(h)}{\log(1-\frac{1}{h})} \cdot \left(-\frac{1}{4}\log\left(1 - \frac{1}{h}\right)\right)\right)\right)^{2h-2} \\
&= \left(1 - \exp\left(-\ln(h)\right)\right)^{2h-2} \\
&= \left(1 - \frac{1}{h}\right)^{2h-2} \\
&\geq (2e)^{-2}.
\end{aligned}
$$

Multiplying these lower bounds gives us the proposed probability for $\Pr\left[Y_{h-1} = 1\right]$. ■

Lemma 4.3 states that a path from s to t that goes via each hub $V_0 \dots, V_h$ achieves a stretch of 2^{h-1}. By Theorem 4.10, we can choose the parameters ℓ, δ_L and r such that this event happens with constant probability if we assume that the initial prefix bit distance is d_0^*. Naturally, the goal is now to choose h as large as possible compared to the number of vertices in the graph. First, we determine an upper bound for the total number of vertices in the constructed graph, depending on h.

Lemma 4.11: *The graph consists of at most*

$$\frac{\ln(2) \cdot N \cdot 2^{h \cdot b^*}}{2^{d_0^*-1}} + 3\ell \cdot 2^{h-1} + \frac{N}{2^{d_0^*-1}} - 1$$

vertices. The parameters $b^* = \log\left(\frac{r}{\ln(2)}\right)$ *and* ℓ *are chosen as in Theorem 4.10.*

Proof. We determine the number of vertices in the hubs $V_1, \dots, V_{h-1}$ and the number of vertices in the paths in P and Q separately. For the sum over all $n_i = \ln(2) \cdot N \cdot 2^{i \cdot b^* - d_0^* + 1}$, we get the following upper bound:

$$\sum_{i=1}^{h-1} n_i = \frac{\ln(2) \cdot N}{2^{d_0^*-1}} \cdot \sum_{i=1}^{h-1} 2^{i \cdot b^*} = \frac{\ln(2) \cdot N}{2^{d_0^*-1}} \cdot \left(\frac{2^{h \cdot b^*} - 1}{2^{b^*} - 1} - 1\right) \leq \frac{\ln(2) \cdot N \cdot 2^{h \cdot b^*}}{2^{d_0^*-1}}.$$

With $|V(P_i)| = 2^i \cdot \ell - 1$ for $i \in \{0, \dots, h-1\}$ and $|V(Q_i)| = 2^i \cdot \ell$ for $i \in \{1, \dots, h-2\}$ and $|V(Q_0)| = \ell + 1$, we get for the number of vertices in the paths:

$$\begin{aligned}
\sum_{i=0}^{h-1} |V(P_i)| + \sum_{i=0}^{h-2} |V(Q_i)| &= \sum_{i=0}^{h-1} \left(2^i \cdot \ell - 1\right) + 1 + \sum_{i=0}^{h-2} 2^i \cdot \ell \\
&= \left(2^h - 1\right) \cdot \ell - (h-1) + 1 + \left(2^{h-1} - 1\right) \cdot \ell \\
&= \left(3 \cdot 2^{h-1} - 2\right) \cdot \ell - h - 2 \\
&\leq 3\ell \cdot 2^{h-1} - 2.
\end{aligned}$$

Including the s and the vertices in V_h, there are

$$1 + n_h + \sum_{i=1}^{h-1} n_i + \sum_{i=0}^{h-1} |V(P_i)| + \sum_{i=0}^{h-2} |V(Q_i)| = \frac{\ln(2) \cdot N \cdot 2^{h \cdot b^*}}{2^{d_0^*-1}} + 3 \cdot 2^{h-1} \cdot \ell + \frac{N}{2^{d_0^*-1}} - 1$$

vertices in the graph. ∎

We now choose a specific value for the initial prefix bit distance d_0^*, and show that the resulting graph consists of at most 2^{2^h} vertices, where h is the number of hops. Thus, we achieve a stretch of $\Theta(\log(n))$, where n is the number of vertices in the constructed graph.

Theorem 4.12: *Let* $d_0^* = \log(N) - h - \log(h)$ *for some* $h \in \mathbb{N}$. *Let* ℓ, δ_L *and* r *be as in Theorem 4.10. The probability that there is a vertex* $t \in V_h$ *such that the found path from* s *to* t *has length at least* $2^{h-1} \cdot \mathrm{dist}(s,t)$ *is at least* $(2e)^{-4} \cdot \left(1 - e^{-1}\right)$. *Further, the constructed graph consists of at most* 2^{2^h} *vertices for sufficiently large* h.

Proof. We first show that the number of vertices in the constructed graph using the chosen parameters does not exceed 2^{2^h}. By Lemma 4.11, the number of vertices in the graph is at most

$$|V(G)| \leq \frac{\ln(2) \cdot N \cdot 2^{h \cdot b^*}}{2^{d_0^*-1}} + 3\ell \cdot 2^{h-1} .$$

For the chosen parameters and sufficiently large h, this is at most 2^{2^h}.

Since the parameters ℓ, δ_L and r are chosen as in Theorem 4.10, the probability that $Y_{h-1} = 1$ for a fixed target t with the desired initial bit distance d_0^* is at least $(2e)^{-4}$. By Lemma 4.4, the probability that there is such a vertex $t \in V_h$ with the desired initial prefix bit distance d_0^* is at least $1 - e^{-1}$. This gives us

$$\begin{aligned} \Pr[Y_{h-1}] &= \Pr\left[Y_{h-1} = 1 \;\middle|\; \exists t \in V_h \colon \operatorname{dist}_b(s,t) = d_0^*\right] \cdot \Pr\left[\exists t \in V_h \colon \operatorname{dist}_b(s,t) = d_0^*\right] \\ &\geq \frac{1}{(2e)^4} \cdot \left(1 - \frac{1}{e}\right). \end{aligned}$$

That means that the algorithm routes via V_{h-1} with at least constant probability. From there, the algorithm must take the path P_{h-1} to reach the target vertex t in V_h. In this case, the stretch of the found path is at least 2^{h-1} by Lemma 4.3. ∎

The following corollary summarizes the findings of this chapter.

Corollary 4.13: *There is a graph G with two vertices s and t such that the stretch $S_{s,t}$ of the path found by KIRA is at least $\frac{1}{2}\log(n)$ with constant probability, where n is the number of vertices.*

The construction of the graph depends on multiple parameters, which affect the value of the stretch, as well as the probability with which KIRA finds a path with high stretch. To achieve a higher stretch, the graph needs more hubs since the construction maximizes the stretch asymptotically in regard to the number of hubs by Lemma 4.1. But then, the hubs have to be smaller than before since increasing the number of vertices in the graph would reduce the achieved stretch (compared to the number of vertices). However, as a consequence of smaller hubs, we would have fewer interesting vertices and thus, a smaller probability that KIRA indeed finds a path that passes through all hubs. This gives us a trade-off between the achieved stretch and the probability with which it occurs, which may be investigated further by tuning the parameters.

5 Conclusions

In this thesis, we studied properties of R^2/Kad, a routing protocol on networks implemented in the routing tier of KIRA, from a theoretical perspective.

In particular, we analyzed if and how KIRA is able to establish KIRA-connectivity on arbitrary graphs. First, we proved basic properties of KIRA-connectivity, which behaves quite differently from the usual definition of connectivity on graphs. For instance, we showed that KIRA-connectivity is neither symmetric nor transitive, which complicates analyses. By constructing a graph that is not KIRA-connectable with constant probability, we proved that KIRA is not always able to connect a graph in general. We found that the different mechanisms implemented in the original variant of KIRA play an important role for the connectivity and need to be combined in certain cases to establish KIRA-connectivity.

Further, we considered different variants of KIRA with slightly modified mechanisms, which proved to be useful in certain scenarios. In particular, a variant we proposed can always establish connectivity if a new vertex joins an already KIRA-connected graph. If we assume that vertices have global knowledge of the topology and the ID assignment of the graph, a variant of KIRA is able to connect any overlay graph. However, such an assumption is not applicable in practice, since it is a core concept of the algorithm that vertices only have a limited view of the graph. For paths, we proposed a strategy with modified mechanisms to establish KIRA-connectivity.

In the second part, we studied bounds for the stretch of paths found by KIRA for both deterministic and random ID assignments. For deterministic ID assignments, we showed that the worst case stretch is at most exponential in the prefix bit distance of the two vertices, and that this bound is tight. Using a similar construction, we provided a lower bound for the stretch assuming random ID assignments. More specifically, we proved that there is a graph with two vertices s and t such that the path found by KIRA from s to t has stretch $\Theta(\log(n))$ with constant probability.

5.1 Future Work

For both properties of KIRA that we studied in this thesis, there are still interesting questions that are left unanswered, some of which we provide in the following.

For instance, we showed that there are graphs that are not KIRA-connectable with constant probability by defining a path structure that acts as a separator. However, these examples are small in the sense that they only have constant diameter. A natural attempt to generalize the approach by extending the graphs attached to the separator proved to be not successful since it is possible that such a separator can be bridged over. As a consequence, we conjecture that graphs, where every pair of vertices is contained in a sufficiently large number of distinct paths, are KIRA-connectable with high probability. This has yet to be proven.

Further, investigating more variants of KIRA may lead to efficient routing strategies, which are provably successful in establishing connectivity with high probability or even deterministically. More specifically, generalizing the strategy that we proposed for paths to arbitrary

graphs looks promising if we modify the closest-k-response by diversifying the contacts the final overlay hop responds with. To develop other strategies, it might be necessary to further modify the mechanisms already implemented in KIRA, and it would be useful to deepen the analysis of the effects of each mechanism on the KIRA-connectivity in more detail.

By constructing a family of graphs where KIRA finds a path with high stretch, we determined a lower bound for the stretch. This construction depends on multiple parameters, which may be tuned to achieve even higher stretch, possibly with a lower probability, enabling a trade-off between stretch and probability. It would be interesting to further evaluate this trade-off and additionally, obtain an upper bound on the stretch that is achieved with constant or even high probability.

Although establishing connectivity and finding short paths are important goals of most routing protocols, there are other qualities that get increasingly important with increasing size and complexity of a network. Examples include robustness against any kind of failures by providing good alternative paths, or how well network traffic is distributed across the network. On the evaluated graphs, KIRA seems to be robust and shows no indication of traffic concentrations [BZDH22]. However, this has not yet been analyzed from a theoretical perspective.

Bibliography

[BS07] Andreas Binzenhöfer and Holger Schnabel. "Improving the Performance and Robustness of Kademlia-Based Overlay Networks". In: *Kommunikation in Verteilten Systemen (KiVS), 15. Fachtagung Kommunikation in Verteilten Systemen, KiVS 2007, Bern, Schweiz, 26. Februar - 2. März 2007*. Edited by Torsten Braun, Georg Carle, and Burkhard Stiller. Springer, 2007, pp. 15–26. DOI: *10.1007/978-3-540-69962-0_2*.

[BZDH22] Roland Bless, Martina Zitterbart, Zoran Despotovic, and Artur Hecker. "KIRA: Distributed Scalable ID-based Routing with Fast Forwarding". In: *IFIP Networking Conference, IFIP Networking 2022, Catania, Italy, June 13-16, 2022*. IEEE, 2022, pp. 1–9. DOI: *10.23919/IFIPNetworking55013.2022.9829816*.

[Cae+06] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony I. T. Rowstron. "Virtual ring routing: network routing inspired by DHTs". In: *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pisa, Italy, September 11-15, 2006*. Edited by Luigi Rizzo, Thomas E. Anderson, and Nick McKeown. ACM, 2006, pp. 351–362. DOI: *10.1145/1159913.1159954*.

[CD13] Xing Shi Cai and Luc Devroye. "A Probabilistic Analysis of Kademlia Networks". In: *Algorithms and Computation - 24th International Symposium, ISAAC 2013, Hong Kong, China, December 16-18, 2013, Proceedings*. Edited by Leizhen Cai, Siu-Wing Cheng, and Tak Wah Lam. Vol. 8283. Springer, 2013, pp. 711–721. DOI: *10.1007/978-3-642-45030-3_66*.

[CD15] Xing Shi Cai and Luc Devroye. "The Analysis of Kademlia for Random IDs". In: *Internet Math.* Volume 11 (2015), pp. 572–587. DOI: *10.1080/15427951.2015.1051674*.

[Che52] Herman Chernoff. "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations". In: *The Annals of Mathematical Statistics* Volume 23 (Oct. 1952), pp. 493–507. DOI: *10.1214/aoms/1177729330*.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*. Vol. 173. Springer, 2012. ISBN: 978-3-642-14278-9.

[ER75] Paul Erdos and Pál Révész. "On the length of the longest head-run". In: *Topics in information theory* Volume 16 (1975), pp. 219–228.

[For04] Bryan Ford. "Unmanaged Internet Protocol: taming the edge network management crisis". In: *Comput. Commun. Rev.* Volume 34 (2004), pp. 93–98. DOI: *10.1145/972374.972391*.

[Hee10] Bernhard Heep. "R/Kademlia: Recursive and topology-aware overlay routing". In: Auckland, New Zealand. Auckland, New Zealand: IEEE, 2010, pp. 102–107. ISBN: 978-1-4244-8171-2. DOI: *10.1109/ATNAC.2010.5680244*.

[HKW17] Henner Heck, Olga Kieselmann, and Arno Wacker. "Evaluating Connection Resilience for the Overlay Network Kademlia". In: *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*. Edited by Kisung Lee and Ling Liu. IEEE Computer Society, 2017, pp. 2581–2584. DOI: *10.1109/ICDCS.2017.101*.

[JCZ11] Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. "VIRO: A scalable, robust and namespace independent virtual Id routing for future networks". In: *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*. IEEE, 2011, pp. 2381–2389. DOI: *10.1109/INFCOM.2011.5935058*.

[KBR08] Joseph S. Kong, Jesse S. A. Bridgewater, and Vwani P. Roychowdhury. "Resilience of structured P2P systems under churn: The reachable component method". In: *Comput. Commun.* Volume 31 (2008), pp. 2109–2123. DOI: *10.1016/j.comcom.2008.01.051*.

[LCW05] Dmitri Loguinov, Juan Casas, and Xiaoming Wang. "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience". In: *IEEE/ACM Trans. Netw.* Volume 13 (2005), pp. 1107–1120. DOI: *10.1109/TNET.2005.857072*.

[Mal+09] Dahlia Malkhi, Siddhartha Sen, Kunal Talwar, Renato Fonseca F. Werneck, and Udi Wieder. "Virtual Ring Routing Trends". In: *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*. Edited by Idit Keidar. Vol. 5805. Springer, 2009, pp. 392–406. DOI: *10.1007/978-3-642-04355-0_42*.

[MM02] Petar Maymounkov and David Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". In: *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*. Edited by Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron. Vol. 2429. Springer, 2002, pp. 53–65. DOI: *10.1007/3-540-45748-8_5*.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. ISBN: 978-0-521-83540-4. DOI: *10.1017/CBO9780511813603*.

[OHKY10] Zhonghong Ou, Erkki Harjula, Otso Kassinen, and Mika Ylianttila. "Performance evaluation of a Kademlia-based communication-oriented P2P system under churn". In: *Comput. Networks* Volume 54 (2010), pp. 689–705. DOI: *10.1016/j.comnet.2009.09.022*.

[Rat+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. "A scalable content-addressable network". In: *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA*. Edited by Rene L. Cruz and George Varghese. ACM, 2001, pp. 161–172. DOI: *10.1145/383059.383072*.

[RSS13] Stefanie Roos, Hani Salah, and Thorsten Strufe. "Comprehending Kademlia Routing - A Theoretical Framework for the Hop Count Distribution". In: (July 2013). arXiv: *1307.7000*.

[Sch90] Mark F. Schilling. “The longest run of heads”. English. In: *The College Mathematics Journal* Volume 21 (1990), pp. 196–207. ISSN: 0746-8342. DOI: *10.2307/2686886*.

[Sin+10] Ankit Singla, Brighten Godfrey, Kevin R. Fall, Gianluca Iannaccone, and Sylvia Ratnasamy. “Scalable routing on flat names”. In: *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology, CoNEXT 2010, Philadelphia, PA, USA, November 30 - December 03, 2010*. Edited by Jaudelice Cavalcante de Oliveira, Maximilian Ott, Timothy G. Griffin, and Muriel Médard. ACM, 2010, p. 20. DOI: *10.1145/1921168.1921195*.

[SR06] Daniel Stutzbach and Reza Rejaie. “Improving Lookup Performance Over a Widely-Deployed DHT”. In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*. IEEE, 2006. DOI: *10.1109/INFOCOM.2006.329*.

[SRS14a] Hani Salah, Stefanie Roos, and Thorsten Strufe. “A Lightweight Approach for Improving the Lookup Performance in Kademlia-type Systems”. In: *CoRR* Volume abs/1408.3079 (2014). arXiv: *1408.3079*.

[SRS14b] Hani Salah, Stefanie Roos, and Thorsten Strufe. “Diversity entails improvement: A new neighbour selection scheme for Kademlia-type systems”. In: *14th IEEE International Conference on Peer-to-Peer Computing, P2P 2014, London, United Kingdom, September 9-11, 2014, Proceedings*. IEEE, 2014, pp. 1–10. DOI: *10.1109/P2P.2014.6934304*.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. *Birthday Paradox for Multi-collisions*. 2006. DOI: *10.1007/11927587_5*.

[Sto+03] Ion Stoica, Robert Tappan Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. “Chord: a scalable peer-to-peer lookup protocol for internet applications”. In: *IEEE/ACM Trans. Netw.* Volume 11 (2003), pp. 17–32. DOI: *10.1109/TNET.2002.808407*.

[WXZ05] Shengquan Wang, Dong Xuan, and Wei Zhao. “Analyzing and enhancing the resilience of structured peer-to-peer systems”. In: *J. Parallel Distributed Comput.* Volume 65 (2005), pp. 207–219. DOI: *10.1016/j.jpdc.2004.09.009*.