TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science, Chair of Privacy and Data Security

Analyzing and Enhancing Routing Protocols for Friend-to-Friend Overlays

Dissertation

Submitted to the Faculty of Computer Science, TU Dresden,
in Partial Fulfillment of the Requirements for the Degree of Dr. rer. nat.

by
Dipl.-Math. **Stefanie Roos**
born on 17 October 1986 in Frankfurt/Main, Germany

First Referee:	Prof. Dr. Thorsten Strufe	TU Dresden, Germany
Second Referee:	Prof. Prateek Mittal, PhD	Princeton University, USA
Fachreferent:	Prof. Dr. Ulrike Baumann	TU Dresden, Germany

Dresden, March 2016

Abstract

The threat of surveillance by governmental and industrial parties is more eminent than ever. As communication moves into the digital domain, the advances in automatic assessment and interpretation of enormous amounts of data enable tracking of millions of people, recording and monitoring their private life with an unprecedented accurateness. The knowledge of such an all-encompassing loss of privacy affects the behavior of individuals, inducing various degrees of (self-)censorship and anxiety. Furthermore, the monopoly of a few large-scale organizations on digital communication enables global censorship and manipulation of public opinion. Thus, the current situation undermines the freedom of speech to a detrimental degree and threatens the foundations of modern society.

Anonymous and censorship-resistant communication systems are hence of utmost importance to circumvent constant surveillance. However, existing systems are highly vulnerable to infiltration and sabotage. In particular, *Sybil attacks*, i.e., powerful parties inserting a large number of fake identities into the system, enable malicious parties to observe and possibly manipulate a large fraction of the communication within the system. Friend-to-friend (F2F) overlays, which restrict direct communication to parties sharing a real-world trust relationship, are a promising countermeasure to Sybil attacks, since the requirement of establishing real-world trust increases the cost of infiltration drastically. Yet, existing F2F overlays suffer from a low performance, are vulnerable to denial-of-service attacks, or fail to provide anonymity.

Our first contribution in this thesis is concerned with an in-depth analysis of the concepts underlying the design of state-of-the-art F2F overlays. In the course of this analysis, we first extend the existing evaluation methods considerably, hence providing tools for both our and future research in the area of F2F overlays and distributed systems in general. Based on the novel methodology, we prove that existing approaches are inherently unable to offer acceptable delays without either requiring exhaustive maintenance costs or enabling denial-of-service attacks and de-anonymization.

Consequentially, our second contribution lies in the design and evaluation of a novel concept for F2F overlays based on insights of the prior in-depth analysis. Our previous analysis has revealed that greedy embeddings allow highly efficient communication in arbitrary connectivity-restricted overlays by addressing participants through coordinates and adapting these coordinates to the overlay structure. However, greedy embeddings in their original form reveal the identity of the communicating parties and fail to provide the necessary resilience in the presence of dynamic and possibly malicious users. Therefore, we present a privacy-preserving communication protocol for greedy embeddings based on anonymous return addresses rather than identifying node coordinates. Furthermore, we enhance the communication's robustness and attack-resistance by using multiple parallel embeddings and alternative algorithms for message delivery. We show that our approach achieves a low communication complexity. By replacing the coordinates with anonymous addresses, we furthermore provably achieve anonymity in the form of plausible deniability against an internal local adversary. Complementary, our simulation study on real-world data indicates that our approach is highly efficient and effectively mitigates the impact of failures as well as powerful denial-of-service attacks. Our fundamental results open new possibilities for anonymous and censorship-resistant applications.

Zusammenfassung

Die Bedrohung der Überwachung durch staatliche oder kommerzielle Stellen ist ein drängendes Problem der modernen Gesellschaft. Heutzutage findet Kommunikation vermehrt über digitale Kanäle statt. Die so verfügbaren Daten über das Kommunikationsverhalten eines Großteils der Bevölkerung in Kombination mit den Möglichkeiten im Bereich der automatisierten Verarbeitung solcher Daten erlauben das großflächige Tracking von Millionen an Personen, deren Privatleben mit noch nie da gewesener Genauigkeit aufgezeichnet und beobachtet werden kann. Das Wissen über diese allumfassende Überwachung verändert das individuelle Verhalten und führt so zu (Selbst-)zensur sowie Ängsten. Des weiteren ermöglicht die Monopolstellung einiger weniger Internetkonzernen globale Zensur und Manipulation der öffentlichen Meinung. Deshalb stellt die momentane Situation eine drastische Einschränkung der Meinungsfreiheit dar und bedroht die Grundfesten der modernen Gesellschaft.

Systeme zur anonymen und zensurresistenten Kommunikation sind daher von ungemeiner Wichtigkeit. Jedoch sind die momentanen System anfällig gegen Sabotage. Insbesondere ermöglichen es Sybil-Angriffe, bei denen ein Angreifer eine große Anzahl an gefälschten Teilnehmern in ein System einschleust und so einen großen Teil der Kommunikation kontrolliert, Kommunikation innerhalb eines solchen Systems zu beobachten und zu manipulieren. F2F Overlays dagegen erlauben nur direkte Kommunikation zwischen Teilnehmern, die eine Vertrauensbeziehung in der realen Welt teilen. Dadurch erschweren F2F Overlays das Eindringen von Angreifern in das System entscheidend und verringern so den Einfluss von Sybil-Angriffen. Allerdings leiden die existierenden F2F Overlays an geringer Leistungsfähigkeit, Anfälligkeit gegen Denial-of-Service Angriffe oder fehlender Anonymität.

Der erste Beitrag dieser Arbeit liegt daher in der fokussierten Analyse der Konzepte, die in den momentanen F2F Overlays zum Einsatz kommen. Im Zuge dieser Arbeit erweitern wir zunächst die existierenden Evaluationsmethoden entscheidend und erarbeiten so Methoden, die Grundlagen für unsere sowie zukünftige Forschung in diesem Bereich bilden. Basierend auf diesen neuen Evaluationsmethoden zeigen wir, dass die existierenden Ansätze grundlegend nicht fähig sind, akzeptable Antwortzeiten bereitzustellen ohne im Zuge dessen enorme Instandhaltungskosten oder Anfälligkeiten gegen Angriffe in Kauf zu nehmen.

Folglich besteht unser zweiter Beitrag in der Entwicklung und Evaluierung eines neuen Konzeptes für F2F Overlays, basierenden auf den Erkenntnissen der vorangehenden Analyse. Insbesondere ergab sich in der vorangehenden Evaluation, dass Greedy Embeddings hoch-effiziente Kommunikation erlauben indem sie Teilnehmer durch Koordinaten adressieren und diese an die Struktur des Overlays anpassen. Jedoch sind Greedy Embeddings in ihrer ursprünglichen Form nicht auf anonyme Kommunikation mit einer dynamischen Teilnehmermengen und potentiellen Angreifern ausgelegt. Daher präsentieren wir ein Privätssphäre-schützenden Kommunikationsprotokoll für F2F Overlays, in dem die identifizierenden Koordinaten durch anonyme Adressen ersetzt werden. Des weiteren erhöhen wir die Resistenz der Kommunikation durch den Einsatz mehrerer Embeddings und alternativer Algorithmen zum Finden von Routen. Wir beweisen, dass unser Ansatz eine geringe Kommunikationskomplexität im Bezug auf die eigentliche Kommunikation sowie die Instandhaltung des Embeddings aufweist. Ferner zeigt unsere Simulationstudie, dass der Ansatz effiziente Kommunikation mit kurzen Antwortszeiten und geringer Instandhaltungskosten erreicht sowie den Einfluss von Ausfälle und Angriffe erfolgreich abschwächt. Unsere grundlegenden Ergebnisse eröffnen neue Möglichkeiten in der Entwicklung anonymer und zensurresistenter Anwendungen.

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. This paper was not previously presented to another examination board and has not been published in this form. I am aware that a false statement entails legal consequences.

Name	City, Date, Signature

Acknowledgments

This thesis would not have been possible without the large number of people who supported me. First of all, I want to thank my adviser Thorsten Strufe who got me interested in the topic of privacy in distributed systems and was always available for questions and discussions. He was essential for all presented contributions, being always patient and critical yet encouraging despite my tendencies to be challenging, troublesome, and just plain stubborn.

I am extremely thankful to Prateek Mittal, who volunteered to review this thesis as the external referee and whose work inspired many of the presented results. Furthermore, I thank Ulrike Baumann for being my Fachreferent and providing useful discussions on the mathematical parts of the thesis.

My wholehearted gratitude goes to all my collaborators, who contributed to the presented results. In particular, I thank Andreas Höfer, who introduced me to greedy embeddings and designed some of the algorithms I build upon. Furthermore, I thank Benjamin Schiller for his help on the analysis of Freenet and for developing GTNA, the graph analysis tool I made use of. I am also grateful to Martin Beck who contributed some essential ideas to the anonymization schemes. In addition, I thank Liang Wang and Jussi Kangasharju for participating in a collaborative evaluation of content addressing in greedy embeddings, Giang Nguyen and Patrick Welzel for their involvement in the simulation studies as well as Christina Heider, Jan-Michael Heller, and Florian Platzer for implementing the measurement studies.

The text of this thesis was furthermore significantly improved as a result of reviews from Stefan Köpsell, Hani Salah, Martin Byrenheid, Martin Beck, Benjamin Schiller, Giang Nguyen, and Thorsten Strufe. Furthermore, Stephan Escher was a great help in designing graphics for my slides. During the last years, I also benefited from the experience of my many coworkers, most importantly (in addition to those named above) Simone Ludwig, Lachezar Krumov, Karsten Weihe, Chris Biemann, Frederik Armknecht, Thomas Paul, Mathias Fischer, Shankar Karuppayah, Daniel Germanus, and Stefan Köpsell. I am extremely grateful for all their help and insightful comments.

Last but not least, I thank my family and friends, in particular my parents Manuela and Bruno Roos.

Contents

1 Introduction **1**
1.1 Requirements . . . 3
1.2 Contributions . . . 4

2 Preliminaries **9**
2.1 Notation . . . 9
2.1.1 Set and Probability Theory . . . 9
2.1.2 Complexity Theory . . . 9
2.1.3 Graph Theory and Analysis . . . 10
2.2 Network Embeddings . . . 11
2.2.1 Concepts . . . 11
2.2.2 Distributed Spanning Tree Construction . . . 12
2.2.3 Embedding Algorithms . . . 13
2.2.4 Improving the Resilience . . . 13
2.3 P2P Overlays . . . 14
2.3.1 Concepts . . . 14
2.3.2 Modeling Dynamics . . . 15
2.3.3 Attacks on P2P overlays . . . 16
2.4 F2F Overlays . . . 17
2.4.1 Motivation . . . 17
2.4.2 Concepts . . . 18
2.4.3 Attacks on F2F overlays . . . 18
2.5 Adversary Model and Assumptions . . . 19
2.5.1 Adversary Goals . . . 20
2.5.2 Knowledge . . . 20
2.5.3 Capabilities . . . 20
2.6 Requirements . . . 21
2.6.1 Functionalities . . . 21
2.6.2 Efficiency and Scalability . . . 22
2.6.3 Robustness and Censorship-Resistance . . . 23
2.6.4 Anonymity and Membership-Concealment . . . 23
2.7 Discussion . . . 23

3 Assessing the State-of-the-Art **25**
3.1 Unstructured Approaches . . . 25
3.1.1 Turtle . . . 26
3.1.2 OneSwarm . . . 26
3.1.3 Summary . . . 27
3.2 Embeddings in Low-dimensional l^p Spaces . . . 27
3.2.1 Low-dimensional l^p Spaces . . . 27
3.2.2 Freenet . . . 27
3.2.3 Alternative Embedding Algorithms . . . 30
3.2.4 Alternative Routing Algorithms . . . 31
3.2.5 General Bounds . . . 32
3.2.6 Summary . . . 32
3.3 DHT-like Overlays . . . 33
3.3.1 GNUnet . . . 33
3.3.2 X-Vine . . . 34
3.3.3 MCON . . . 36

3.3.4 Summary . . . 36
3.4 F2F-aided P2P Systems . . . 36
3.4.1 Sybil Resistance . . . 37
3.4.2 Proxy Selection in Anonymity Services . . . 37
3.4.3 Summary . . . 37
3.5 Discussion . . . 37

4 User Models **39**
4.1 Churn . . . 39
4.1.1 Measurement Methodology . . . 39
4.1.2 Measurement Set-up . . . 40
4.1.3 Churn Data Set . . . 40
4.1.4 Validation and Measurement Overhead . . . 41
4.1.5 Limitations . . . 43
4.2 Social Graphs . . . 43
4.3 Discussion . . . 45

5 Methodology **47**
5.1 Probability Theory . . . 47
5.1.1 Random Variables and Stochastic Processes . . . 47
5.1.2 System Models . . . 49
5.2 Efficiency and Scalability . . . 50
5.2.1 Communication Complexity of Routing Algorithms . . . 50
5.2.2 Communication Complexity of Stabilization Algorithms . . . 54
5.2.3 Routing and Stabilization over Time . . . 56
5.2.4 Simulation . . . 56
5.3 Effect of Churn . . . 58
5.3.1 State-of-the-Art . . . 58
5.3.2 General Methodology . . . 58
5.3.3 Worst-Case Bounds . . . 59
5.3.4 Evaluation . . . 62
5.4 Robustness and Censorship-Resistance . . . 63
5.4.1 General Approach . . . 63
5.4.2 Robustness . . . 65
5.4.3 Censorship-Resistance . . . 65
5.5 Anonymity and Membership-Concealment . . . 66
5.5.1 Anonymity . . . 66
5.5.2 Membership-Concealment . . . 68
5.6 Discussion . . . 68

6 Virtual Overlays **69**
6.1 Statement of Result and Proof Idea . . . 70
6.2 Modeling Virtual Overlays as a Stochastic Process . . . 72
6.2.1 Notation . . . 72
6.2.2 Multidimensional Metrics . . . 74
6.2.3 Assumptions . . . 74
6.3 Theoretical Analysis . . . 75
6.3.1 Basic Properties . . . 75
6.3.2 Fully Determined Overlays . . . 77
6.3.3 Partially Determined Overlays . . . 79
6.4 Simulation . . . 84
6.5 Discussion . . . 84

7 Greedy Embeddings **87**
7.1 Statement of Result and Proof Idea . . . 87
7.2 Notation and System Model . . . 89
7.2.1 Graphs and Embeddings . . . 89
7.2.2 Balanced Content Addressing . . . 89
7.3 Analysis . . . 91
7.4 Algorithm Design . . . 93
7.4.1 Unweighted PIE Embedding . . . 93

7.4.2 PrefixEmbedding . . . 94
7.4.3 PrefixSEmbedding . . . 95
7.4.4 Analysis . . . 97
7.5 Simulation . . . 98
7.5.1 Simulation Model and Set-up . . . 98
7.5.2 Expectations . . . 98
7.5.3 Results . . . 98
7.6 Discussion . . . 99

8 VOUTE **101**
8.1 Idea . . . 101
8.2 Design . . . 103
8.2.1 Tree Construction and Stabilization . . . 103
8.2.2 Embedding and Routing . . . 105
8.2.3 Anonymous Return Addresses . . . 107
8.2.4 Content Storage . . . 110
8.3 Efficiency and Scalability . . . 110
8.3.1 Theoretical Analysis . . . 111
8.3.2 Simulations . . . 114
8.4 Robustness and Censorship-Resistance . . . 117
8.4.1 Attack Strategies . . . 117
8.4.2 Theoretical Analysis . . . 118
8.4.3 Simulations . . . 120
8.5 Anonymity and Membership-Concealment . . . 123
8.5.1 Anonymity . . . 123
8.5.2 Membership-Concealment . . . 124
8.6 Discussion . . . 125

9 Conclusion **129**

Appendices **143**

A Freenet Opennet Neighbor Selection **143**
A.1 Methodology . . . 143
A.2 Set-up and Results . . . 144
A.3 Discussion . . . 144

B Inferring Obfuscated Values in Freenet **147**
B.1 Background . . . 147
B.1.1 Freenet's Anonymous Statistics . . . 147
B.1.2 Bayesian Statistics . . . 148
B.2 Inference Algorithm . . . 148
B.2.1 Algorithm Overview . . . 149
B.2.2 Modeling Obfuscation . . . 149
B.2.3 Global Distribution . . . 150
B.2.4 Likelihood Function . . . 150
B.3 Evaluation . . . 150
B.3.1 Scenario . . . 150
B.3.2 Metrics . . . 151
B.3.3 Data Sets . . . 152
B.3.4 Simulation . . . 152
B.3.5 Measurements . . . 154
B.4 Discussion . . . 155

C Computation of Remaining Online Time **157**

D Topology Aware Keys **159**
D.1 Algorithm Design . . . 159
D.2 Simulation . . . 159
D.3 Discussion . . . 162

E PPP Return Addresses **165**

F Extended Results for VOUTE **167**
F.1 Routing length of *WOT* and *SPI* 167
F.2 Robustness and Censorship-Resistance of *WOT* and *SPI* 168
F.3 Traffic per Node 169
F.4 Impact of q 170

List of Tables

2.1 Requirements and their Conflicts 23

3.1 State-of-the-Art Approaches 38

4.1 Maximal Response Interval in Churn Measurement 41
4.2 Scalar Metrics of Social Graphs 44

5.1 Remaining Online Time R^{ind} fitted to Freenet Churn Data 62

8.1 Requirement Fulfillment in VOUTE 126

B.1 Bandwidth Inference Error 155

D.1 Maximum Load in AS Topologies 161
D.2 Routing Length in AS Topologies 162

F.1 Routing Length Comparison of Exemplary Social Graphs 167

List of Figures

1.1 Concept F2F Overlay . . . 2
1.2 F2F Use Cases . . . 3
1.3 F2F Overlay Designs . . . 5
1.4 Thesis Structure . . . 6

2.1 Rooted Spanning Tree Example . . . 12
2.2 PIE Embedding Example . . . 12
2.3 Network Layer and Overlay . . . 14
2.4 Dissemination of Exemplary Request in P2P overlay . . . 14
2.5 Approaches to P2P Connectivity . . . 18
2.6 Basic Algorithms in F2F Overlays . . . 21

3.1 Kleinberg Model vs. F2F overlays . . . 28
3.2 Variants of Greedy Routing . . . 29
3.3 Freenet Swapping . . . 30
3.4 Concept of Virtual Overlays . . . 33
3.5 Tunnel Construction in X-Vine . . . 35

4.1 Churn Behavior in Freenet . . . 42
4.2 Degree and Shortest Paths Length Distributions . . . 44

5.1 System Model Example . . . 50
5.2 Non-greedy Embeddings . . . 52
5.3 Methods for Determining the Routing Complexity . . . 53
5.4 Method for Determining the Stabilization Complexity . . . 56
5.5 Routing and Stabilization Complexity over Time . . . 56
5.6 Simulation-based Evaluation of the Routing Complexity . . . 57
5.7 Session Length of Random Online Node . . . 60
5.8 Online Time Distributions . . . 60
5.9 Delays in Single-Threaded Recursive Routing . . . 61
5.10 Probability of Topology Change Impacting Regular Actions . . . 63
5.11 Simulation-based Evaluation of Disruption Tolerance . . . 65
5.12 Identification by External Data Sources . . . 68

6.1 Concept of Virtual Overlays (Recaptured) . . . 69
6.2 Effect of Tunnel Concatenation . . . 70
6.3 State Information in Virtual Overlays . . . 73
6.4 Tunnel Shortcuts . . . 73
6.5 Fully Determined and Partially Determined Overlays . . . 73
6.6 Mean Tunnel Length in Virtual Overlays . . . 84

7.1 Content Addressing in Tree-based Embeddings . . . 88
7.2 Balanced Content Addressing in Tree-based Embeddings . . . 88
7.3 Closest Nodes to Content . . . 91
7.4 Adverse Scenarios for Balanced Content Addressing in Greedy Embeddings . . . 92
7.5 Illustration of PIE and PrefixEmbedding . . . 94
7.6 Fraction Keys Assigned to Nodes: Algorithm Comparison . . . 99
7.7 Fraction Keys Assigned to Nodes: Topology Comparison . . . 99

8.1 Layers of VOUTE . . . 102

8.2 Original PIE vs. modified PIE . . . 105
8.3 Distances in Tree-based Embeddings . . . 106
8.4 Routing Length in VOUTE . . . 115
8.5 Tree Depth and Stabilization Overhead in VOUTE . . . 116
8.6 Illustration of Resistance Proof . . . 119
8.7 Robustness to Failures in VOUTE . . . 121
8.8 Censorship-Resistance of VOUTE . . . 122
8.9 Attack Resistance . . . 122
8.10 Illustration of Anonymity Proof . . . 123

9.1 F2F Use Cases: Solutions . . . 130
9.2 Attack Strategies with Topology Knowledge . . . 131

A.1 Distance and Degree Distributions in Freenet . . . 144

B.1 Inferred Global Bandwidth Distribution in Freenet . . . 151
B.2 Success of Inferring Bandwidth in Freenet . . . 152
B.3 Error and Certainty in Inference of Bandwidth . . . 154

D.1 Load Distribution in AS Topologies . . . 162

F.1 Robustness: *FB* vs. *WOT* . . . 168
F.2 Censorship-Resistance: *WOT* vs. *SPI* . . . 169
F.3 Traffic per Node . . . 170
F.4 Impact of Parent Acceptance Probability q . . . 170

List of Algorithms

7.1 computePrefixEmbedding() . . . 95
7.2 nextHop(BitSequence key, Node u) . . . 95
7.3 computePrefixSEmbedding() . . . 96
7.4 nextHopS(BitSequence key, Node u) . . . 96
8.1 constructTreeRound() . . . 104
8.2 route() . . . 106
8.3 generateRP() . . . 107
D.1 computeKey(BitSequence f) . . . 160
E.1 addPPPLayer() . . . 166

Chapter 1

Introduction

In the last decades, digital communication has become an integral part of our life. At the same time, large-scale Internet surveillance through governmental and commercial parties has emerged as a serious threat to user privacy. Two of the most prominent examples illustrating the drastic extent of this threat are the unveiling of the NSA's global surveillance program [13] and Facebook's accidental publication of private data on a large scale [4]. The threat presented by accidental data loss is immediate, as normally uninvolved and uninformed parties can abuse the private information for undesired user profiling and criminal purposes such as burglary [9]. In contrast, the impact of global surveillance on users' everyday life is less palpable but all the more dangerous for society as a whole. Large-scale surveillance opens the door for global censorship, in particular the repression of minorities and inconvenient opinions. At worst, companies and governmental parties can abuse their knowledge of and power over enormous amounts of information to manipulate public opinions. In this manner, they can possibly even dictate the results of elections or similar important political decisions. Even without the actual execution of this capability, the knowledge of surveillance on its own calls forth self-censorship [148] and radical personality changes [23]. As a consequence, the current large-scale Internet surveillance drastically undermines freedom of speech, an essential human right and the foundation of modern democracy.

A key issue enabling large-scale surveillance is the convergence towards monopolization of Internet services, so that a handful of companies control the majority of Internet traffic [20]. By restricting the majority of the communication to a small number of parties, large-scale censorship merely requires the cooperation of these companies by sabotage, blackmail, or the application of legal (but potentially immoral) means. Particular examples of the latter are the National Security Letters of the United States [12] and the recent data retention laws ('Vorratsdatenspeicherung') [22] in Germany. The centralized nature of today's Internet service thus enables global surveillance and manipulation at a (relatively, if adversary is in a position of power) low cost. Hence, decentralization of services is the first step in the prevention of global surveillance.

Decentralized systems are either based on distributed servers, e.g., the social network Diaspora [2], or a completely decentralized P2P overlay, e.g., the file-sharing system BitTorrent [1]. While decentralized servers merely distribute the service on several, seemingly independent service providers, P2P networks consists of end devices of everyday users participating as both servers and consumers. In this manner, P2P overlays completely avoid the use of dedicated servers.

However, simply removing the central point of trust only mitigates global surveillance but fails to prevent powerful adversaries from obtaining large amounts of data. Adversaries with sufficient resources, including governmental organizations as well as large companies, are able to observe and possibly manipulate a large fraction of the communication by controlling a large number of the participating servers or end devices, referred to as *nodes*. As a consequence, infiltrating the system with a large number of artificially created nodes controlled by a single entity, commonly referred to as *Sybils*, is generally easy, as for instance recent attacks on the Tor system show [21].

In addition to being vulnerable to Sybil attacks by computationally powerful adversaries, distributed systems introduce additional vulnerabilities by enabling everyday users to directly influence the provided service. Being both consumers and service providers, users have a higher impact on the system and in particular more opportunities to cause injury to the system, intentionally or unintentionally. For example, some vulnerabilities in a subroutine of the anonymous P2P network Freenet allow a single user to undermine the complete service without the need of Sybils or unusual computational power [64]. Furthermore, end devices reveal their network addresses, e.g., IPs, to many not necessarily trustworthy participants. As a consequence, the need to establish connections with untrusted strangers and thus reveal private information or at least the participation in the system to such parties induces numerous vulnerabilities and limits the desire of users to participate in the system. So, merely decentralizing the

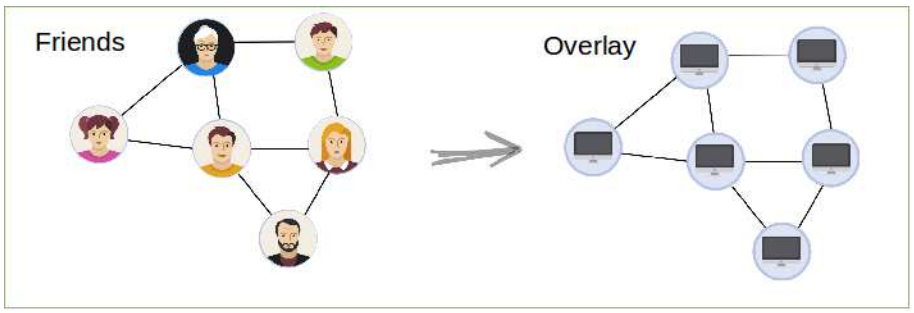


Figure 1.1: Concept Friend-to-Friend overlay: Overlay connections corresponds to trust relations

service does not entail privacy-preserving and censorship-resistant communication.

Friend-to-Friend (F2F) overlays or F2F networks, also called *Darknets* and first suggested in [122], are a highly promising P2P-based approach to overcome the above concerns. Note that the decisive factor governing the impact of Sybil attacks are the number of connections to honest devices, not the number of fake identities. Hence, rather than impeding the creation of fake identities, F2F overlays raise the cost for establishing connections to honest devices. For this purpose, they restrict direct communication to mutually trusted parties. In other words, the connections in a F2F overlay correspond to real-world trust relationships, as illustrated in Figure 1.1. In order to surveil or sabotage the F2F overlay, an adversary has to establish trust relationships through the use of social engineering. We assume that establishing trust relationships is costly, at least in comparison to establishing fake identities in an automated fashion. Thus, if the algorithms of the overlay are secure in the sense that they do not permit nodes with a low connectivity to sabotage the complete system, F2F overlays limit the impact of Sybil attacks. In addition to limiting the impact of Sybil attacks, F2F overlays abolish the need to directly communicate with untrusted strangers, thus providing membership-concealment towards untrusted participants.

By restricting the communication to trusted parties, F2F overlays offer a promising communication substrate for privacy-preserving applications such as email, instant-messaging (between trusted parties as well as between anonymous untrusted parties), file-sharing, social networking, and publish-subscribe. Three exemplary use cases for F2F overlays are the following, illustrated in Figure 1.2:

a) Alice is a whistle blower in an oppressive regime and wants to contact the journalist Bob while hiding the fact that she is the sender of the message from Bob and all other participants.

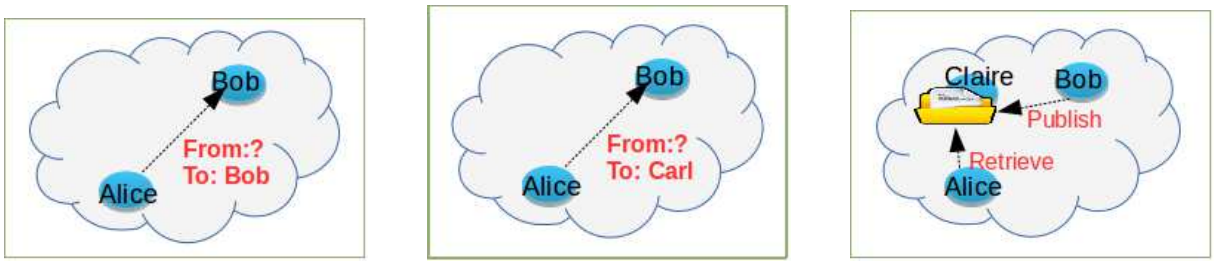
b) Bob has published information about a health issue he is suffering from under a pseudonym. Alice, who is unsure if she is affected by the same condition, wants to contact Bob anonymously for additional information knowing Bob's pseudonym only.

c) Bob has published information regarding the organization of a demonstration without revealing his identity. The information is stored by Claire, who does not want to reveal that she stores the information due to her fear of persecution. Alice wants to access the information without revealing her identity.

So, F2F overlays need to provide anonymous messaging between two untrusted parties and anonymous sharing of content. On a more abstract level, node and content discovery are the two essential required functionalities. In order to attract a sufficient amount of users, efficient realization of the two functionalities is essential, as is robustness to failures and resistance to attacks on the availability.

A number of deployed F2F overlays are available for use, e.g., Turtle [122], OneSwarm [15], GNUnet [65], and Freenet [48]. However, those overlays are poorly analyzed and exhibit high latency, bad quality of service as well as various security issues and privacy leaks [64, 160, 123, 132]. In contrast, there are mainly academic projects, e.g., MCON [160] and XVine [110], which share similar goals. Both MCON and XVine achieve sufficient robustness to failures as well as certain malicious behavior. However, anonymization of communicating untrusted parties is not explicitly considered in both approaches. Furthermore, the communication complexity of messaging and content sharing scales with $\mathcal{O}(\log^2 n)$ for an overlay with n participants while the shortest path in the overlay between any two participants scales with $\mathcal{O}(\log n)$. Thus, neither MCON nor XVine are asymptotically optimal, i.e., their asymptotic communication complexity exceeds the asymptotic lower bound on the communication complexity given by the shortest path between nodes. In addition, the trade-off between the stabilization complexity when nodes join and leave the overlay and the communication complexity of messaging over an extended period of time has not been considered. In summary, the current state-of-the-art leaves room for improvement both with regard to conceptual evaluation and the performance of the algorithms.

In this thesis, we aim to design a F2F overlay that realizes the above functionalities based on censorship-resistant and efficient protocols. We mainly aim to evaluate the underlying concepts of both



(a) (b) (c)

Figure 1.2: Exemplary use cases for F2F overlays: a) messaging with anonymous sender, b) messaging with two anonymous parties using a pseudonym for the receiver, c) anonymous content sharing storing the published content at an otherwise uninvolved user; anonymity here refers to hiding one's identity from communication partners as well as all remaining nodes in the overlay

state-of-the-art approaches and our own designs rather than providing a ready-to-use prototype. For this purpose, we first perform an in-depth evaluation of the state-of-the-art. We show that the existing approaches are inherently unable to simultaneously achieve both efficiency and censorship-resistance. In the second part of the thesis, we design algorithms for communication in F2F overlays. Our design is based upon greedy embeddings, which assign coordinates to nodes in order to facilitate efficient node discovery. However, greedy embeddings have been designed for static environments without malicious parties and the need to hide the identity of participating nodes. Thus, we increase the robustness to failures as well as the resistance to denial-of-service attacks by considering alternative routes and applying multiple diverse greedy embeddings in parallel. Furthermore, we modify the embedding and messaging algorithm to rely on anonymous addresses rather than identifying coordinates. In order to provide content sharing, we establish a virtual overlay on top of the greedy embedding. In this manner, we realize the desired functionalities using efficient and resilient protocols.

We evaluate our algorithms predominately through theoretical analysis, combined with a simplified simulation study. More precisely, we show that the messaging and the stabilization complexity scales logarithmically with the number of participating nodes n. Content sharing is slightly more costly, requiring polylog complexity. Furthermore, we prove that our anonymous addresses indeed always provide sender and receiver anonymity, preventing the attacker from uniquely identifying sender and receiver even if it controls neighbors of both parties. Our extensive simulation study indicates that our algorithms allow faster node discovery and stabilization in the presence of joining and departing nodes than the state-of-the-art approaches. Moreover, the overhead for content sharing only slightly exceeds the overhead of the best state-of-the-art algorithm, while VOUTE requires considerably less stabilization overhead. Last, the robustness to failures and the resistance to various denial-of-service attacks is improved. In summary, our initial design satisfies our requirements and thus presents a promising approach for a real-world implementation.

The contributions presented in this thesis resulted in publications at several top conferences and journals, including INFOCOM [130, 136, 137] and PETS [134]. We adapt the publications for this thesis, with the permission of the corresponding co-authors. In order to provide a more extensive overview of this thesis, we now give a concise but informal summary of our requirements and contributions.

1.1 Requirements

Providing both anonymous messaging as well as content sharing in a F2F overlay is a challenging problem. A F2F overlay has to satisfy many partially conflicting requirements regarding i) *scalability and efficiency*, ii) *robustness and censorship-resistance*, and iii) *anonymity and membership-concealment*. In the following, we motivate each of these three aspects.

Efficiency and Scalability: Efficiency and scalability are key requirements for any large-scale communication system, because people are unlikely to participate if the system does not provide the expected quality-of-service. In particular, efficient communication implies low latencies, fast stabilization after topology changes, and a low communication overhead measured in the number of messages exchanged for messaging, content sharing, and stabilization after node joins and departures. Scalability implies that the communication complexity increases slowly with the number of participants n, indicating that an approach provides efficient service for large user groups, which we might be unable to simulate or observe in real-world systems. As current F2F overlays have been shown to be slow and unreliable [160],

increasing the performance of F2F overlays while still achieving the certain protection goals is the focus of this thesis.

Robustness and Censorship-Resistance: As F2F overlays are dynamic systems with malicious or malfunctioning nodes, they have to be able to provide reliable communication despite topology changes, in particular node departures, and attacks. Here, we define robustness as the ability of a system to function despite failures of individual nodes. Similarly, we define censorship-resistance as the ability of system to function despite malicious nodes aiming to censor communication. The success ratio of the routing, i.e., the fraction of paths between source and destination in the same component that can be found, should decrease *gracefully* with the number of failed nodes or the number of edges between honest and malicious nodes. We aim to improve the robustness to failures and the resistance to censorship in comparison to state-of-the-art approaches.

Anonymity and Membership-concealment: As stated above, our goal is to prevent an adversary from identifying participants without establishing a direct overlay connection. As a consequence, we even demand that the anonymized topology of the social graph is not revealed, as individuals can be identified from anonymized graphs [115]. In addition, an adversary should never be able to uniquely identify the sender or receiver of a message.

We have now summarized our requirements in sufficient detail to understand the contributions of this thesis. Note that we offer a more formal definition in Section 2 after introducing some basic notation and concepts.

1.2 Contributions

Overall, our contributions regarding the design of F2F overlays can be classified into three parts:

1. *Preparation*: Review of state-of-the-art, attainment of realistic data sets for our simulation study, and methodology development
2. *State-of-the-Art Evaluation*: In-depth evaluation of existing concepts that have not been sufficiently analyzed in existing literature
3. *Own Design and Evaluation*: Design and evaluation of algorithms for a novel F2F overlay concept based on the insights of the previous parts

Together, these parts present a thorough analysis of F2F overlays culminating in the design of a novel promising concept for messaging and content sharing in such overlays. In addition to these contributions regarding principal questions of F2F overlays, this thesis also influenced contributions in related fields. In the following, we give a more detailed overview of the three parts as well as the related contributions.

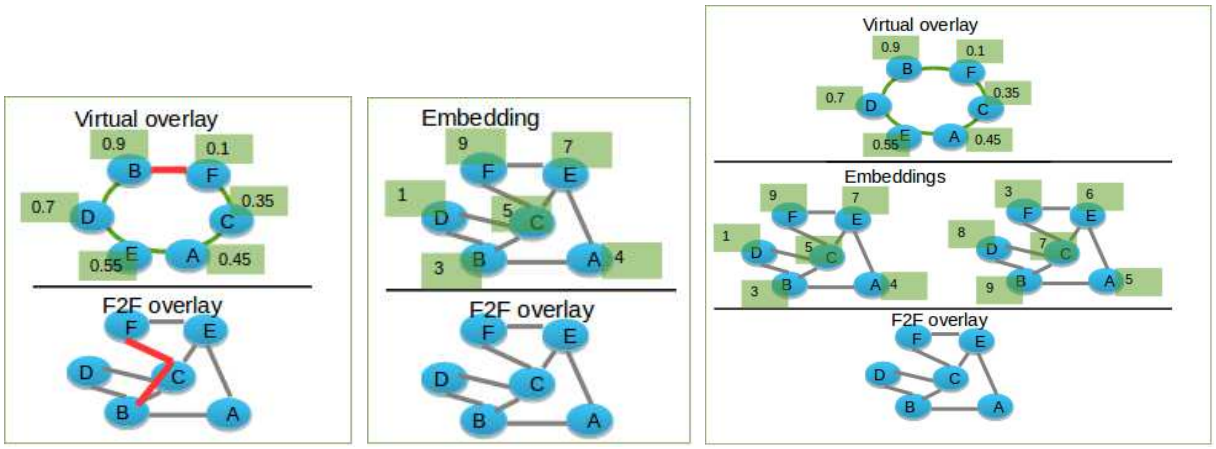
Preparation: The preparation consists of identifying gaps in the state-of-the-art and developing the appropriate tools for our evaluation. Before considering our contributions, we introduce some preliminaries in Chapter 2.

In Chapter 3, we review the state-of-the-art and categorize the existing approaches into i) unstructured overlays, ii) virtual overlays, and iii) (network) embeddings. While unstructured approaches cannot achieve the required efficiency, it remains unclear if the virtual overlays and embeddings are suitable approaches. Thus, we conclude that we require a detailed analysis of these approaches.

An appropriate evaluation consists of a rigorous scientific methodology applied to realistic (user) models. In Chapter 4, we introduce user models for F2F overlays. In our evaluation, we require i) social graphs as models for the connections in the F2F overlay and ii) join and departure pattern of users in a F2F overlay in order to model the dynamics in the user group. While we found suitable existing data sets for social graphs, we performed a measurement study in Freenet to obtain the required join and departure patterns. The measurement study has been published in PETs 2014 [134].

Afterwards, we specify our methodology in Chapter 5. Our methodology for evaluating F2F overlays both theoretical and by simulation leverages the standard methodology for evaluating distributed systems. However, we adapt the methodology for F2F overlays and extend it slightly in order to address issues such as the trade-off analysis of stabilization and node discovery complexity. These extensions are of general interest and are partially published in [131].

The development of users models and evaluation methodology enables the in-depth evaluation in the remainder of the thesis.



(a) Virtual Overlays (b) Embeddings (c) VOUTE

Figure 1.3: F2F overlays approaches: a) Virtual overlays such as [110, 160] establish a routable structure and connecting neighboring nodes in that structure through *tunnels* in the F2F overlay, e.g. red tunnel between B and F, b) network embeddings such as the Freenet swapping algorithm [48] assign coordinates that facilitate routing by forwarding to neighbors with coordinates close to destination, c) VOUTE [130] combines the two by facilitating messaging between nodes through multiple network embeddings and enabling content sharing in a virtual overlay leveraging the messaging protocol in the embeddings for communication between virtual neighbors

State-of-the-Art Evaluation: Our review of the state-of-the-art reveals that virtual overlays and embeddings have not been sufficiently evaluated to determine if they fulfill our requirements. Hence, we perform an in-depth analysis based on our previously developed methodology.

First, in Chapter 6, we show that virtual overlays offer the desired functionalities but fail to fulfill all requirements simultaneously. More precisely, virtual overlays aim to achieve a routable structure by assigning random coordinates to nodes. Neighboring nodes in the routable structure then have to establish *tunnels*, paths in the F2F overlay, to communicate indirectly. The concept is illustrated in Figure 1.3a. We show that maintaining tunnels of a sufficiently short length requires exceedingly high stabilization complexity in the presence of joining and departing nodes. The result has been published in INFOCOM 2015 [137].

Second, we show that embeddings in their current form cannot combine efficiency and resistance to censorship. Embeddings assign coordinates to nodes in order to facilitate messaging between non-neighboring nodes. More precisely, the embedding coordinates take the role of the receiver address in the message. In addition, content sharing is realized by assigning coordinates to content and storing content on nodes with similar coordinates. We illustrate the concept of embeddings in Figure 1.3b. In previous work, we showed that purely local round-based embeddings into Euclidean space fail to achieve the necessary efficiency [135, 136, 138]. In Chapter 7, we focus on greedy tree-based embeddings. Here, the coordinates are assigned based on a rooted spanning tree of the underlying social graph of the F2F overlay. However, spanning trees do not allow for efficient censorship-resistant content sharing without requiring costly stabilization schemes. The result has been partially published in MobiArch 2014 [139].

Thus, our analysis shows that the current concepts are inherently unable to satisfy our requirements.

Own Design and Evaluation: In Chapter 8, we present the design and evaluation of Virtual Overlays Using Tree Embeddings (VOUTE). Our design relies on tree-based embeddings for messaging between nodes. For content sharing, we establish an additional virtual overlay using the coordinates of the embeddings for communication rather than tunnels. Figure 1.3c exemplary illustrates the main idea of our design. We modified the standard greedy embedding scheme in order to achieve all our requirements, as detailed in the following.

The scalability and efficiency of messaging and content sharing follows from the efficiency of greedy embedding and virtual overlays. Greedy embeddings allow for efficient stabilization, and the removal of tunnels reduces the stabilization complexity of virtual overlays. We prove that the overall communication complexity increases at most polylog with the number of participants. Furthermore, we perform a simulation study to show that we indeed achieve an improved performance in comparison to the state-of-the-art.

Robustness and censorship-resistance require several modifications due to the vulnerability of the tree structure underlying the embeddings. For this purpose, we use multiple embeddings, allow backtracking if

a message cannot be forwarded anymore, and optionally include additional nodes in the communication. In our theoretical analysis, we prove that the changes i) do not considerably reduce the scalability, and ii) indeed increase robustness and censorship-resistance in comparison to the unmodified scheme. In our simulation study, we quantify the improvement in comparison to both the unmodified scheme and state-of-the-art approaches.

We need to modify embeddings in order to provide anonymity and membership-concealment. Using the coordinates of nodes as addresses allows identifying the receivers of messages. Furthermore, the coordinates reveal essential information about the structure of the social graph, which can be used to identify participants. In order to prevent the identification of nodes, we first modify the nature of the assigned coordinates such that guessing a node's coordinate is no longer efficiently possible. Second, we propose a protocol for encrypting coordinates such that the encrypted coordinates represent anonymous addresses. Nodes can thus communicate without requiring the actual receiver coordinates. We show that the modification indeed achieves sender and receiver anonymity at the price of a slightly increased local computation complexity while maintaining the same communication complexity.

In summary, VOUTE achieves all our requirements on a conceptual level. The results have been accepted for publication in INFOCOM 2016 [130].

Collaborations: While I am the primary originator of the presented work, it is the result of extensive discussions with my supervisor Thorsten Strufe and my coworkers. In particular, Andreas Höfer contributed to the design of tree-based embedding algorithms in Chapter 7. Benjamin Schiller contributed ideas for the Freenet measurement study in Chapter 4 and developed the graph analysis tool GTNA [144] used for the simulation study. Martin Beck is partially responsible for the anonymous return address generation algorithm presented in Chapter 8. Liang Wang and Jussi Kangasharju contributed to the evaluation of content addressing schemes for greedy embeddings, presented in Chapter 7. Furthermore, Jan-Michael Heller, Christina Heider, Stefan Hacker, and Florian Platzer helped in conducting measurements in Freenet. I utilize the results in agreement with the remaining contributors. In adherence to the fact that the presented work is the result of collaborations, I will use the pronoun 'we' rather than 'I' throughout the thesis.

Related Contributions: In addition to the results presented in the main part of the thesis, we evaluated various related questions in the area of F2F overlays and anonymous P2P-based communication. As these results are not directly related to the conceptual analysis of F2F overlays, we present them in the appendix.

Appendix A and Appendix B present vulnerabilities of the Freenet code. During our measurement study, we discovered these vulnerabilities and subsequently designed alternative protocols. The improved protocols and their evaluation are published in PETs 2014 [134] and NetSys 2015 [132]. Furthermore, they have been integrated in the current implementation of Freenet.

Appendix C gives a detailed account of mathematical transformations necessary to obtain results in Chapter 5. We excluded the details of the derivation in order to focus on the underlying ideas and improve the readability of the chapter.

Furthermore, Appendix D presents an algorithm for content sharing in tree embeddings without the need of virtual overlays. However, the algorithm is unsuitable for F2F overlays due its high stabilization complexity and need for global knowledge of the overlay topology. However, the algorithm is of interest for other areas such as content-centric networking and has been published in this context [139].

In Appendix E, we present an additional layer of encryption for coordinates in VOUTE. Extended simulation results concerning VOUTE are presented in Appendix F.

In addition to the results presented in the appendix, we apply our novel evaluation techniques in the areas of large-scale discovery services [72, 133, 141], Botnets [85] and P2P live-streaming [116].

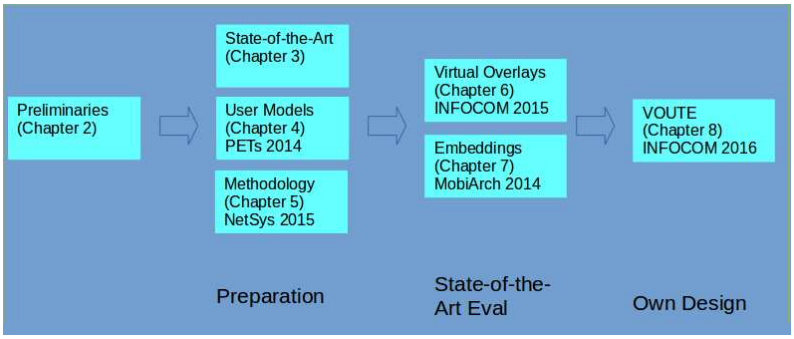


Figure 1.4: Thesis structure and corresponding publications

In the following, we present the decisive progress in the area of F2F overlays achieved by

1. Developing evaluation methods for F2F overlays,
2. Providing an in-depth analysis of the state-of-the-art, and
3. Designing and evaluating a novel conceptual approach.

We recapitulate the individual contributions and the resulting structure of the thesis in Figure 1.4.

Chapter 2

Preliminaries

In this chapter, we present necessary concepts and derive our requirements. We first introduce some elementary notation and concepts from the area of graph theory. Then, we present two possibilities for structuring graphs such that short routes between nodes can be found efficiently. The first possibility is the assignment of meaningful coordinates to nodes that mirror the structure of the graphs. We introduce such so called *network embeddings* in Section 2.2. The second possibility to structure graphs lies in the assignment of random coordinates and an adaption of the connections between nodes. We introduce this approach in the context of P2P overlays in general, as structured overlays are based upon this principle. In addition to summarizing approaches for route discovery in P2P overlays, Section 2.3 identifies potential attacks on P2P overlays. Afterwards, we consider F2F overlays in Section 2.4, focusing if and to what extent F2F overlays are vulnerable to the previously identified attacks. In Section 2.5 and Section 2.6, we formalize our adversary model and specify our requirements based on the insights gained from the previously presented concepts.

2.1 Notation

We now give a short overview of our most important general notation, corresponding to common notation in the areas of set, probability, complexity, and graph theory.

2.1.1 Set and Probability Theory

Basic set theory is used to express relations between objects. A *set* A is a unsorted collection of arbitrary *elements*. In this thesis, sets are usually indicated by curly braces, i.e., a set A containing the elements a_0 and a_1 is defined by $A = \{a_0, a_1\}$. Furthermore, we write $a \in A$ if a is contained in the set A, and $a \notin A$ if a is not contained in A. We denote the *union* of two sets A_1 and A_2 by $A_1 \cup A_2$, their *intersection* by $A_1 \cap A_2$, and the *difference* of A_1 and A_2 by $A_1 \setminus A_2$. The complement of a set A is denoted by $A^{\perp}$. Set theory is the basis for expressing a multitude of mathematical concepts.

Throughout this thesis, we make use of probabilistic user models and algorithms. Thus, our results usually only hold with a certain probability or in expectation. Hence, we denote the *probability* of an event E by $P(E)$. In particular, the probability of a *random variable* X to attain a value in a set A is denoted by $P(X \in A)$. Furthermore, the *expectation* of X is $\mathbb{E}(X)$. In the course of this thesis, we leverage additional concepts and results from probability theory for our theoretical analysis. As these concepts are very specific to our methodology, we introduce them at a later point.

2.1.2 Complexity Theory

The cost of an algorithm is usually expressed in asymptotic bounds, i.e., by characterizing the growth of costs in terms of the input size disregarding constants and terms of a lower order. We here formally define the "bigO"-notation and related asymptotic bounds, which are used throughout this thesis. In general, such asymptotic bounds define the behavior of functions for large inputs. In this manner, a function of the generic cost in terms of some input parameter of an algorithm is related to simpler well-understood function.

Commonly, the complexity of an algorithm consists of i) *computation complexity*, the number of elementary operations required to execute the algorithm, ii) *storage complexity*, the amount of bits required to store the necessary information to execute the algorithm, and iii) *communication complexity* or *overhead*, the number of messages sent between nodes to execute a distributed algorithm. Our focus in this

thesis is the communication complexity, since it is the limiting factor for the performance of the overall system.

Asymptotic bounds can refer to the maximum or average/expected costs of an algorithm. As our system model is probabilistic, we are mostly interested in upper bounds on the expected costs.

For the formal definition of the asymptotic growth in terms of the input size n, let $(x_i)_{i\in\mathbb{N}}$ be a sequence of positive real numbers. We define the limes superior $limsup_{i\to\infty}x_i$ as the number r such that for all $\epsilon > 0$, there exists n_0 such that $x_i \leq r + \epsilon$ for all $i > n_0$. In other words, $limsup_{i\to\infty}x_i$ is an asymptotic upper bound on the sequence $(x_i)_{i\in\mathbb{N}}$. Now, consider two functions $f, g : \mathbb{N} \to \mathbb{N}$ with

$$\eta = limsup_{i\to\infty}\frac{f(n)}{g(n)} \qquad \in \mathbb{R}_+ \cup \{\infty\}.$$

In terms of the above motivation, f denotes the costs of the algorithm and g the simplified function. Based on the above notation, we write

- $f = \mathcal{O}(g)$ if $\eta < \infty$, i.e, g grows asymptotically at least as fast as f,
- $f = \Omega(g)$ if $\eta > 0$, i.e., g grows asymptotically at most as fast f,
- $f = o(g)$ if $\eta = 0$, i.e, g grows asymptotically faster than f,
- $f = \omega(g)$ if $\eta = \infty$, i.e., g grows asymptotically slower than f, and
- $f = \theta(g)$ if $f = \mathcal{O}(g)$ and $f = \Omega(g)$, i.e., f grows asymptotically at the same rate as g.

In this manner, we can relate the communication complexity f_A of a distributed algorithm A in terms of the number of nodes n to a function g, e.g., $f_A = \mathcal{O}(\log n)$ indicates that the cost scales at most logarithmically with the network size.

2.1.3 Graph Theory and Analysis

Graph theory is an important tool to describe the properties of a distributed system. In the following, we first define the concept of a graph, followed by important metrics for analyzing graphs. Afterwards, we present selected concepts specific to this thesis, namely properties of social networks and spanning trees.

A graph $G = (V, E)$ consists of a set of *nodes* or *vertices* V and a set of *edges* or *links* $E \subset V \times V$. Graphs are natural representations of both computer networks, with nodes representing devices and edges connections, as well as social networks, with nodes representing individuals and edges personal relationships. Throughout this thesis, we focus on bidirectional graphs, i.e., graphs G, so that $(u, v) \in E$ iff $(v, u) \in E$, mirroring the requirement of a mutual trust relationship in a F2F overlay. The neighbors of a node v are denoted by $N(v) = \{u \in V : (v, u) \in E\}$.

In the context of communication in a distributed system, the notion of a *path* is of particular importance. We define a path p as a vector $p = (v_0, \ldots, v_l)$ of nodes $v_i \in V$ and $(v_i, v_{i+1}) \in E$ for all $i = 0, \ldots, l-1$ and $l > 0$. The number l giving the edges of a path is referred to as the *length* of the path, and v_0 and v_l are called the *start-* and *endpoint*, respectively. A graph is divided into *connected components* such that two nodes are in the same component if and only if there is a path from v to u. Being in the same connected component is a requirement for two nodes to communicate with each other.

Two metrics for comparing graphs, which we use throughout the thesis to explain certain results, are the *degree distribution* and the *shortest path length distribution*. Note that the degree $deg(v) = |N(v)|$ of a node v is defined as the number of neighbors of v. As a consequence, the degree distribution Deg of a graph G corresponds to the distribution of degrees within the set of nodes, i.e.,

$$P(Deg = i) = \frac{|\{v \in V : deg(v) = i\}|}{|V|}. \tag{2.1}$$

The shortest path length $sp(u, v)$ between two nodes v and u is defined as the minimum length of all paths with startpoint v and endpoint u. We also refer to the length of the shortest paths of two nodes as their *hop distance* or the *number of hops*. Consequently, the *shortest path length distribution* SP of the graph G is defined by the fraction of pairs of distinct nodes with a certain shortest path length, i.e.,

$$P(SP = i) = \frac{|\{v, u \in V : u \neq v, sp(v, u) = i\}|}{|V|(|V| - 1)}. \tag{2.2}$$

The *diameter* $diam$ of a graph G is defined as the longest shortest path over all pairs of nodes,

$$diam = \max_{u,v\in V, u\neq v} sp(u, v). \tag{2.3}$$

The shortest paths present a lower bound on the number of messages required for two nodes to communicate.

F2F overlays are defined by the trust relations of its participants. Graphs representing social relationships between humans are called *social graphs* or *social networks*. We rely on two experimentally validated assumptions about social networks [41], which can be expressed in terms of the above metrics. First, many analyzed social networks exhibit a *scale-free* degree distribution, i.e., we have

$$P(Deg = i) = \theta\left(\frac{1}{i^\alpha}\right), \text{ for } 2 < \alpha < 3. \tag{2.4}$$

Equation 2.4 implies that social networks contain many nodes with a low degree and very few nodes with a high degree. Furthermore, the average degree, i.e., the expected value of the random variable defined by Equation 2.4, is bound by a constant independent of the size of the social network. The second property associated with social networks is the existence of short paths between all nodes. We hence assume that the diameter is bound logarithmically in the network size n, i.e.,

$$diam = \mathcal{O}(\log n). \tag{2.5}$$

A logarithmic diameter indicates that the bound on the communication costs between two nodes is at least logarithmic as well.

In the following, we provide some background in preparation for the next section on greedy embeddings. Greedy embeddings heavily rely on *rooted spanning trees*, connected subgraphs $ST = (V, E^T)$ of $G = (V, E)$ such that $|E^T| = |V| - 1$, $E^T \subset E$, and a distinguished node r called the *root* of ST. Nodes in the spanning tree are described based on their relation to the root. In particular, the *level* or *depth* of a node u is given by the length of the path from r to u in T. The *depth* or *height* of a tree is defined as the maximal depth of a node in the tree. If u is not the root, the *parent* of u is defined as the neighbor v in T with a shorter path to r than u (restricted to paths in T, i.e., traversing only of edges in the spanning tree), whereas the remaining neighbors in T are u's *children*. A node u is called a *descendant* of a node v if v is contained on the path between u and r in T. If u is a descendant of v, v is called an *ancestor* of u. A node without children is called a *leaf*, whereas nodes with children are called *internal nodes*. Figure 2.1 illustrates some of the presented concepts. In the following, we explain the concept of (tree-based) greedy embeddings based on the presented notation.

2.2 Network Embeddings

Network embeddings assign *coordinates* or *addresses* to nodes in a fixed graph. In this manner, they identify nodes by these coordinates and thus allow *routing* requests from a source node s to a destination e based on e's coordinate. In general, node coordinates should be assigned in such a way that the routing has a low communication complexity. Embeddings thus present an interesting approach with F2F overlays in mind because they offer low communication complexity without the need of establishing additional connections. We start by introducing the basic concepts in that field of research, followed by specific algorithms including approaches that aim to increase the robustness to failures.

2.2.1 Concepts

Let $G = (V, E)$ be a connected finite graph. A *network embedding* $id : V \to \mathbf{X}$ for a metric space $(\mathbf{X}, \delta_X)$ with a distance function δ_X assigns *coordinates* to nodes. Thus, any coordinate assignment is an embedding by definition. However, in general, coordinates are supposed to facilitate e.g., efficient routing. For this purpose, the concept of *greedy embeddings* is central. The embedding id is called *greedy* if any node s given a *destination coordinate* $id(e)$ other than its own has at least one neighbor whose coordinate is closer to the destination, i.e., as defined in [118],

$$\forall s, e \in V, s \neq e, \exists v \in N(s), \delta_X(id(v), id(e)) < \delta_X(id(v), id(s)). \tag{2.6}$$

As a consequence, nodes can forward a request addressed to $id(e)$ to their neighbor with the closest coordinate. At some point the request is guaranteed to reach e. The distance to the destination coordinate decreases in each step, i.e., is a decreasing process without *local minima*. We define this stateless distributed algorithm of always forwarding to the neighbor closest to the destination as *greedy routing*.

Now, we shortly discuss the general idea for constructing a greedy embedding. Though there exists a multitude of greedy *embedding algorithms*, they all follow the same four abstract steps:

1. Construct a rooted spanning tree ST.

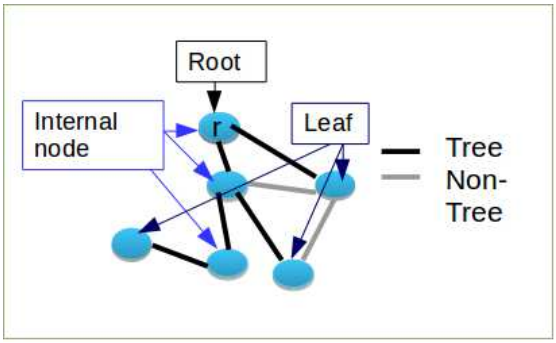


Figure 2.1: Example of a rooted spanning trees, including root, internal nodes, and leaves

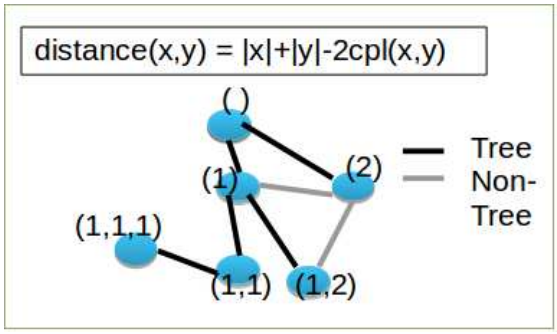


Figure 2.2: Example of the PIE embedding $|x|$: length of coordinate x, $cpl(x, y)$: common prefix length of two coordinates

2. Each internal node in ST enumerates its children.

3. The root receives a predefined coordinate.

4. Children derive their coordinate from the parent's coordinate and the enumeration index assigned by the parent (e.g., [87, 56, 18, 80]).

The coordinates are assigned such that the embedding of the spanning tree is greedy. In a distributed version of the above algorithm, the root node r computes its coordinate and the coordinates of its children. Afterwards, r sends messages to all its children containing their respective coordinates. Upon receiving their coordinates, internal nodes compute the coordinates of their children and send them to the children. The algorithm terminates if all leaves have coordinates.

The communication complexity of the algorithm consists of the complexity of i) establishing the spanning tree and ii) distributing the coordinates. Regardless of the algorithms for spanning tree construction and coordinate generation, the number of messages for coordinate assignment are $n - 1$, the number of edges in a tree.

Subsequent to the coordinate assignment, nodes can route based on the receiver's coordinate. During routing, a forwarding node v selects the neighbor whose coordinate is closest to the receiver's coordinate. All neighbors in the graph are considered for forwarding decisions, not only the neighbors in the tree. Forwarding via a non-tree edge, which we will call a *shortcut*, allows for a faster reduction of the distance and so usually reduces the length of the discovered routes. Overall, the communication complexity of routing corresponds to at most twice the height of tree, i.e., the length of the longest path between two nodes in the tree. In particular, a logarithmic diameter indicates that the communication complexity is $\mathcal{O}(\log n)$ as well.

Stabilization of greedy embeddings is required whenever nodes join or leave, i.e., some or all coordinates are re-assigned such that the new embedding containing additional or less nodes is still greedy. Various embeddings [56, 18, 80] can react to such dynamics without renewing the complete embedding. New nodes join the trees as leaves. So, joining requires only a constant overhead, namely contacting one neighbor as the new parent that assigns the new coordinate in turn. If any node leaves, the children have to reconnect and all descendants have to be reassigned coordinates. Only if the root leaves, the spanning tree and the embedding have to be re-established entirely. While the existing algorithms are hence designed to stabilize without complete re-computation, the complexity of the local stabilization algorithm is yet to be explicitly analyzed.

Having introduced the common approach of tree-based greedy embeddings, we now detail specific algorithms for i) distributed spanning tree construction and ii) coordinate assignment.

2.2.2 Distributed Spanning Tree Construction

There exist various approaches for the construction of spanning trees in distributed, possibly dynamic environments. They either first select the root followed by the construction of the tree starting from the root or execute root and tree construction in parallel. The authors of the PIE embedding [80] suggest the use of the standard algorithm for spanning tree construction, designed by Radia Perlman in the mid-80s [120]. Each node randomly selects a random number, called *ID* in the following, and periodically sends the highest *ID* known to him to its neighbors together with the length of the shortest known path to the corresponding node. As long as a node u is unaware of any *ID* higher than its own, u considers itself the root of the (partly established) tree. Otherwise, u chooses one of its neighbors with the shortest path

to the node with the highest locally known *ID* as its parent. In this manner, trees are first established locally and then merged into one large tree. Note that if the tree is not supposed to be a tree of minimum depth, the above algorithm can be modified to use a different parent selection.

Now, we give a short overview of the complexity of root election and spanning tree construction when using the Perlman algorithm. The number of messages required until the spanning tree is stable are closely related to the diameter *diam* of the graph, because the information of the highest *ID* has to be spread to all nodes. As a result, it takes *diam* rounds of period updates until all nodes are aware of the correct root node. Assuming that each node sends one message for each update, the communication complexity of the algorithm is $\mathcal{O}(n \log n)$ messages for a graph with diameter $\mathcal{O}(\log n)$.

However, Perlman's root election of the above algorithm is not secure against attacks, because it permits nodes to either fake a ID and become the influential root or to constantly inject new higher IDs. A secure leader election protocol such as [98, 151] might be used to select the root. However, their complexity is at least $\Omega(n^2)$, because all nodes need to send messages to each other. For a dynamic environment with the frequent need for reconstruction due to a departed root node, such a communication complexity might be unacceptable high.

2.2.3 Embedding Algorithms

Though embedding algorithms generally rely on a spanning tree and assign coordinates according to the tree structure, the nature of the assigned coordinates is highly diverse: Embeddings into hyperbolic space such as [56, 18, 87] allow embedding in low-dimensional spaces, which has been shown to be impossible in Euclidean space [103]. However, proposed hyperbolic embeddings are extremely complex and do not scale with regard to the number of bits required for coordinate representation [18]. Custom-metric approaches have been designed to overcome these shortcomings.

For instance, the PIE [80] embedding is a custom-metric embedding, which encodes a node's position in the spanning tree as an integer-valued vector of varying length. So, PIE first assigns an empty vector as the coordinate to the root node r. In an unweighted graph, child coordinates are then derived from the parent coordinate by concatenating the parent coordinate with the index assigned to the child by the parent. In this manner, a node v's coordinate represents the route from r to v. Consequently, the distance corresponds to the shortest path between two nodes in the tree, which is not necessarily identical with their shortest path in the complete graph. An example of the PIE embedding for a small graph with a spanning tree of height 3 is displayed in Figure 2.2.

2.2.4 Improving the Resilience

If nodes fail, the embedding is no longer greedy and routing tends to fail due to local minima in the distance function. There are two approaches to mitigate the resilience to such failures by either modifying the routing algorithm or constructing multiple parallel embeddings.

Cvetkovski and Crovella [56] suggest a modified routing algorithm for dealing with local minima in the distance. Nodes keep track on the number of times they have received a request. Rather than terminating in the absence of a closer neighbor, a node u considers the set of neighbors that have received the request the least times. From this set, u selects the neighbor closest to the destination. The algorithm guarantees the discovery of a path to the destination but might require a linear communication complexity. Furthermore, it fails to consider attackers dropping messages.

In contrast, Herzen et al. suggest a variant of multiple embeddings [80]. Apart from a global embedding, nodes establish local embeddings containing a subset of nodes, all sharing the same metric space (M, δ). Each embedding has a unique identifier i and a node u participating in embedding i has coordinate $id_i(u)$. Let $em(u)$ be the set of identifiers of all embeddings u participates in. Given the set $em(e)$ and the corresponding coordinates of a destination e, a node forwards the request to the neighbor v with the closest coordinate in any embeddings, i.e., the node v such that

$$\Delta(v) = \min_{i \in em(e) \cap em(v)} \delta(id_i(v), id_i(e))$$

is minimal. The possibility to choose from multiple embeddings entails shorter routes and increases the success ratio if the embedding is not greedy due to node failures. However, as the algorithm is single-threaded, i.e., the request is only forwarded to one node in each step, it does not adequately deal with attackers dropping messages.

In summary, greedy embeddings are a promising tool for routing requests from one node to another. It remains to see if they can be integrated in F2F overlays, in particular with regard to robustness, censorship-resistance, and anonymization. Nevertheless, the existence of algorithm allowing low communication complexity for messaging indicates that we can indeed design such algorithms for F2F overlays.

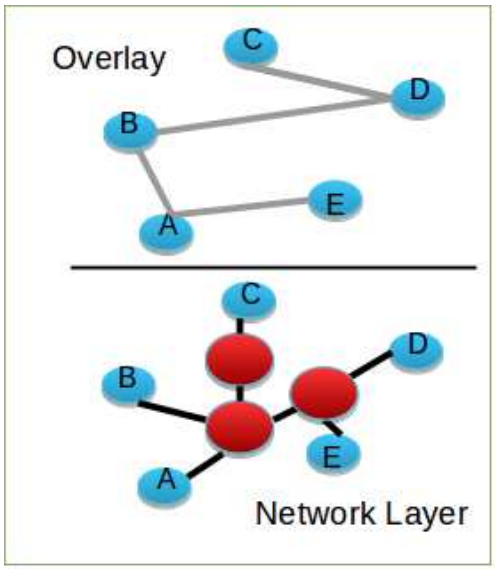


Figure 2.3: Exemplary underlay and overlay consisting of end devices (blue) and routers (red) with physical connections (black) and overlay connections (gray)

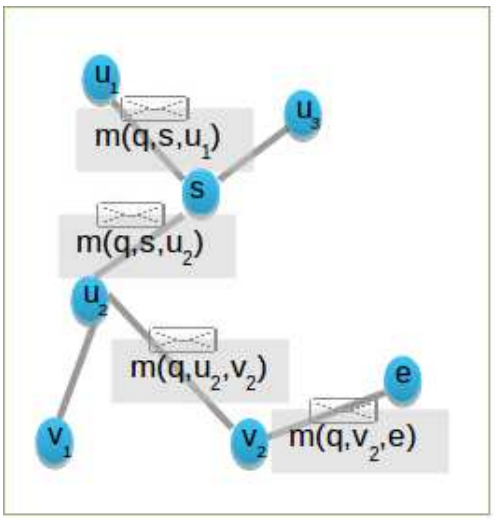


Figure 2.4: Dissemination of exemplary request q in a P2P overlay. The initiator s forwards messages $m(q, s, u_1)$ and $m(q, s, u_2)$ to two of its neighbors, u_2 forwards the request to its neighbor v_2, and finally the request reaches the receiver e.

2.3 P2P Overlays

P2P overlays, also called *P2P systems* or *P2P networks*, have emerged as an alternative to the common client-server architecture of the Internet. In P2P overlays, nodes participate as both consumers and servers. The predominant application is file-sharing, as illustrated by the popularity of file-sharing applications with several millions of users such as Napster [42], BitTorrent [161], and eMule [156]. In addition to providing file-sharing, P2P overlays are promising communication substrates for social networking [55], video streaming [79], and instant messaging [31] among others. In this section, we first define the elementary concepts of P2P overlays, followed by a short overview on models for user behavior in P2P overlays. Due to the constant joining and departing of nodes, such models are essential for evaluating algorithms under realistic conditions. Last, we summarize common attacks on P2P overlays. Our notation follows [62], which provides an in-depth overview of P2P overlays.

2.3.1 Concepts

Generally, *users* or *participants* refer to humans whose end devices are part of a system and execute the respective software. In contrast, *nodes* refer to the actual devices executing the software. Though we mostly follow this terminology throughout the thesis, we usually abbreviate 'nodes corresponding to users with a mutual trust relationship' by *trusted nodes*.

An overlay represents a subgraph of the fully connected mesh of all devices participating in a system. In this manner, an *overlay* abstracts from the physical connections on the *underlay*. More precisely, let $\mathbf{U} = (V_U, E_U)$ be graph of routers and connected end devices and their physical links. The underlay provides the necessary protocols, such as IP lookups, for communication between any two end devices. Thus, P2P overlays generally abstract from the message delivery process in the underlay and consider a path between two nodes in the underlay as one link. As it is generally impossible to keep track of all nodes in the P2P overlay, each node v explicitly maintains a set of connections to a small subset of the other nodes, v's *(overlay) neighbors* $N(v)$. So, we represent an overlay by the graph $\mathbf{O} = (V, E)$, also called the *(overlay) topology*, with $V \subset V_U$ denoting the set of devices participating in the P2P overlay and E denoting the connections that nodes explicitly maintain. Nodes that are not overlay neighbors communicate by forwarding requests via multiple overlay connections. We illustrate the relation of underlay and overlay in Figure 2.3. Our research is concerned with designing protocols for communication within the overlay independent of the underlay, so that we disregard the nature of the underlay connections throughout this thesis.

On an abstract level, P2P overlays provide a service as follows: The sender s of a request q selects a set of overlay neighbors $U(q) \subset N(s)$. For each of these nodes $u \in U(q)$, s generates a message $m(q, s, u)$, which s sends to u. Upon receiving the message $m(q, s, u)$, u executes the necessary actions. Commonly

these actions result in sending additional messages $m(q, u, v)$ to neighbors $v \in N(u)$. In this manner, the request is distributed within the network until a termination criterion applies. Possible termination criteria are the successful delivery to an intended receiver e or the abortion due to a maximal number of forwarding operations. In the first case, we call the path, along which a request is forwarded from a node s to a node e the *route* from s to e. Figure 2.4 displays an exemplary dissemination of a request q.

There are various criteria to classify P2P overlays. In the following, we classify overlays based on the principles underlying their communication protocols. In other words, our classification considers the algorithms determining how nodes forward requests. Here, we differentiate between *unstructured*, *hierarchical* and *structured* overlays. In addition, there are *hybrid* overlays. We now shortly summarize the idea of each class and give examples.

Unstructured overlays are characterized by their in-deterministic neighbor selection and use of non-directional searching. More precisely, nodes forward requests to a set of neighbors chosen independently of the requests' content. In this manner, unstructured overlay achieve a low stabilization complexity without the need to maintain a certain structure. However, locating individual nodes requires sending a request to potentially all nodes in the overlay. So, unstructured overlays are commonly used if a high fraction of nodes can supply the desired service. For instance, file-sharing with highly popular files replicated vastly throughout the network is a typical application of unstructured overlays, because locating a service provider does not require an exhaustive search. A real-world example for an unstructured overlay is Gnutella [128], which floods requests to all nodes within a certain number of hops of the requester. Gnutella has been shown to generate a lot of traffic [25].

Hierarchical overlays divide the nodes into classes based on their computation power or reliability. Commonly, there are two classes: *super-peers* with higher than average computational power or reliability and *normal peers*. The super-peers establish an (unstructured) overlay on their own with normal peers connecting to super-peers only. Normal peers forward their requests to their super-peers. Super-peers can collectively locate a peer (super or normal) to serve these requests. In this manner, hierarchical overlays follow a similar principle as distributed servers, except for the fact that super-peers are not necessarily required to be constantly online. A real-world example for a hierarchical overlay is KaZaA, which had more than 3 million users a day in 2003 [74]. Due to the importance of the super-peers, hierarchical overlays suffer from the same lack of censorship-resistance as distributed servers: A take-down of the super-peers effectively sabotages the service [96].

Structured overlays establish connections between nodes in order to establish a *routable structure* and forward requests only to a small set of nodes leveraging the nature of the structure to satisfy a request. The predominant concept for structured overlays are *distributed hash tables* (DHTs). A hash table is a data structure storing objects based on an object key, usually the hash of the binary representation of the object. DHTs distribute such a key-value storage over all nodes in an overlay. More precisely, they provide the functionality of a hash table, namely storing or retrieving a value or object via its key, in a distributed system. For this purpose, nodes are assigned *coordinates* from the same metric space as the keys and store values whose keys close to this coordinate. Furthermore, in order to allow for efficient and scalable discovery of values, the overlay topology is structured such that nodes only need to maintain $\mathcal{O}(\log n)$ connections to discover values by sending $\mathcal{O}(\log n)$ messages. In contrast to unstructured overlays, DHTs allow efficient delivery of requests but have a higher stabilization complexity. Departing nodes destroy the overlay structure and thus require repair mechanisms. During the last 15 years, a multitude of DHTs have been suggested. The most widely known approaches are Chord [154], one of the oldest and most intensively researched designs, and Kademlia [104], the most widely applied distributed discovery service implemented e.g., in the eMule client [3] and as part of the BitTorrent protocol [1]. Overall, structured overlays are much more scalable than unstructured or hierarchical overlays but require additional stabilization. However, the selective adaption of the overlay topology assumes that overlay connections can be established between any pair of nodes.

In addition to the above three classes, there are hybrid overlays, which make use of a central server for resource discovery. Two famous examples are the tracker in the original BitTorrent and the central index in Napster [42]. As motivated in the introduction of this thesis, central entities invite censorship, so that we do not consider any overlays relying on such an entity.

This completes our overview of basic concepts in the area of P2P overlays. In particular, we have reviewed four classes of overlays. As a result, we found that structured overlays achieve high efficiency and scalability, but are not directly applicable to F2F overlays. Nevertheless, similar concepts can be applied.

2.3.2 Modeling Dynamics

P2P overlays are dynamic systems in the sense that nodes join and depart constantly. In this manner, the topology of the overlay changes over time. The *churn rate*, i.e., the frequency of such *topology changes*,

is essential for the choice of routing and stabilization algorithms because a higher churn rate requires more frequent stabilization and thus implies an increased need for efficient stabilization. A *churn model* describes the probability distributions governing the topology changes.

The *session length*, i.e., the time a node typical remains within the overlay between a join and a departure, is of particular importance for the algorithm design. Short sessions indicate that the stabilization complexity should be low, because the topology changes frequently and nodes are unlikely to profit much from the structure gained by the stabilization. In contrast, long sessions indicate that structured overlays indeed achieve a lower overall communication complexity because the reduced complexity for frequent resource discovery outweighs the comparable high stabilization complexity.

In real-world measurements, four patterns of session length distributions have been observed. More precisely, the measured data has been fitted to the following density functions f_S:

- Exponential [76], i.e., $f_S(x) = \frac{1}{\lambda} e^{-x/\lambda}$ for $\lambda > 0$
- Pareto [39], i.e., $f_S(x) = k \frac{\lambda^k}{(x+\lambda)^{k+1}}$ for $k, \lambda > 0$
- Weibull [157], i.e., $f_S(x) = k \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$ for $k, \lambda > 0$
- Lognormal [157], i.e., $f_S(x) = \frac{k}{\sqrt{2\pi}x} e^{-k^2 \ln(x/\lambda)/2}$ for $k, \lambda > 0$

The results indicate that the user behavior varies with the type of overlay and the offered service. As none of the measurements targets F2F overlay or similar privacy-preserving services, we require a specific measurement study to obtain a realistic user model.

In addition to the session length, some studies consider the inter-session length and the connectivity factor as well. The inter-session length corresponds to the time users typically remain offline between two consecutive sessions. In contrast, the connectivity factor denotes the fraction of the measurement period a node is online, independently of the number of sessions it remains online. Both measures can only be obtained if identities are persistent, i.e., nodes can be re-identified after rejoin.

2.3.3 Attacks on P2P overlays

In general, everyone can join a P2P overlay. As a consequence, it is easy for an attacker to infiltrate the overlay. The attacker can then launch attacks on either the availability of the service or the privacy of its users. While the implementation of the attacks is specific to the actual protocol of the overlay, many attacks follows similar principles. In the area of P2P overlays, research is mainly concerned with attackers that are

1. internal, i.e., the attacker is participating in the overlay rather than being an outsider
2. local, i.e., the attackers are only aware of their neighbors and the messages it receives, and
3. active, i.e., the attacker manipulates protocols by dropping, manipulating, faking, and replaying messages.

In the following, we describe the most common principle attack ideas.

Sybil Attack: A Sybil attack is actually the preparation of an actual attack. The adversary inserts a large number of nodes in the overlay in order to gain many connections and intercept a large fraction of the requests.

Routing Table Poisoning Attack: Similar to Sybil attacks, Routing Table Poisoning attacks such as [152] are mainly a preparation of a follow-up attack. They are complementary to Sybil attacks in the sense that the attacker manipulates the protocol such that a large number of nodes establish connections to its node(s). In this manner, the attacker again intercepts a large fraction of requests.

Black Hole/Eclipse Attack: A Black Hole or Eclipse attack [150] is a denial-of-service attack building on a Routing Table Poisoning Attack. The attacker nodes drop all requests they receive, thus resulting censoring communication indiscriminately.

Localized Eclipse Attack: Complementary to Eclipse attacks, localized Eclipse attacks [72] only censor specific requests, e.g., requests to provide certain content. For this purpose, the attacker places nodes such that they are likely to receive such requests. Distributed hash tables are particularly vulnerably to Localized Eclipse attacks because attackers simply have to choose the coordinates of their nodes close to the key of the content they aim to censor.

Pollution and Index Poisoning: During a pollution attack, the attacker generates a large fraction of fake requests or fake content. As for an Eclipse attack, the goal is to indiscriminately censor communication as legitimate requests or content are disregarded due the congestion created by the fake requests. Similarly, index poisoning pollutes the index of file keys rather than the actual files, thus preventing a node from obtaining the key for a file [96].

De-anonymization: The above attacks are mostly concerned with denial-of-service attacks, aiming to disrupt the service. Attacks on the anonymity of users can be prepared by Sybil attacks and Routing Table Poisoning as well. However, the main attack is usually specific to the anonymization technique, e.g., the attack [145] on the ShadowWalker protocol [109]. Thus, de-anonymization attacks cannot be generalized easily.

In this section, we have introduced the basic concepts in the area of P2P overlays. Now, we give a more detailed overview of F2F overlays in the light of the above concepts. In particular, we discuss the applicability of the above attacks on F2F overlays.

2.4 F2F Overlays

In this section, we describe the motivation and the most important concepts of F2F overlays. For the time being, we do not introduce specific approaches, which we consider in our state-of-the-art review in the next chapter.

2.4.1 Motivation

In general, it is not possible to guarantee a reliable and anonymous delivery of a request if the majority of the connections in an overlay are controlled by an attacker. The reason for the lack of resistance against denial-of-service attacks is the fact that two honest nodes require the existence of an overlay path without any adversarial nodes in order to send a request from one node to another. Similarly, anonymity usually requires at least one honest node on each path between sender and receiver, e.g., when using mixes [43] or onion routing [125]. Thus, mitigating attacks requires reducing the likelihood that attackers are on the *route* between sender and receiver. Such mitigation schemes usually consist of two steps:

1. Reduce number of connections between attackers and honest nodes, and
2. Select routes between a source s and a destination e such that connections to attackers are not considerably more likely to be traversed as connections to honest nodes

In summary, the idea is to mitigate attacks by limiting the number of connections to attackers in the overlay and the importance of each connection. While the second step is very specific to the application and the protocols for forwarding requests, the first step of limiting the attacker's connectivity is mostly protocol-independent. Hence, we focus on the question of how to reduce the attacker's connections to honest nodes in the following.

Using the terminology from Section 2.3.3, reducing the number of overlay connections between honest and malicious nodes requires the prevention of both Sybil and Routing Table Poisoning attacks. Without additional protection schemes, P2P overlays are vulnerable to Sybil and Routing Table Poisoning attacks because a single entity can easily gain control over a large fraction of overlay connections by i) inserting a large number of fake nodes and ii) establishing connections to many honest nodes. There are two principle approaches to undermine such attacks. The first approach is to protect against Sybil and Routing Table Poisoning attacks individually by preventing attackers from inserting fake nodes and limiting the degree per node. In contrast, the second approach does not prevent an attacker from establishing fake nodes but aims to restrict the total number of connections of these nodes to honest nodes. Figure 2.5 illustrates the desired impact of the approaches in comparison to an unprotected overlay.

There are multiple approaches to prevent Sybil and Routing Table Poisoning attacks individually. In particular, mitigation strategies for Sybil attacks include i) allowing only one node per unique identifying attribute such as IP address or social security number [60], ii) challenges such as CAPTCHAs supposedly requiring human input [77], and iii) Sybil detection schemes such as [58] that identify Sybils in the trust graph of the overlay users because of their low number of connections to the remaining graph. Approaches to mitigate Routing Table Poisoning include i) frequent changes of neighbors [53] and ii) ensuring a limited number of connections per node by anonymous auditing [150]. Apart from their individual weaknesses, the above approaches all create additional overhead and enforce revealing the membership in the overlay to untrusted strangers.

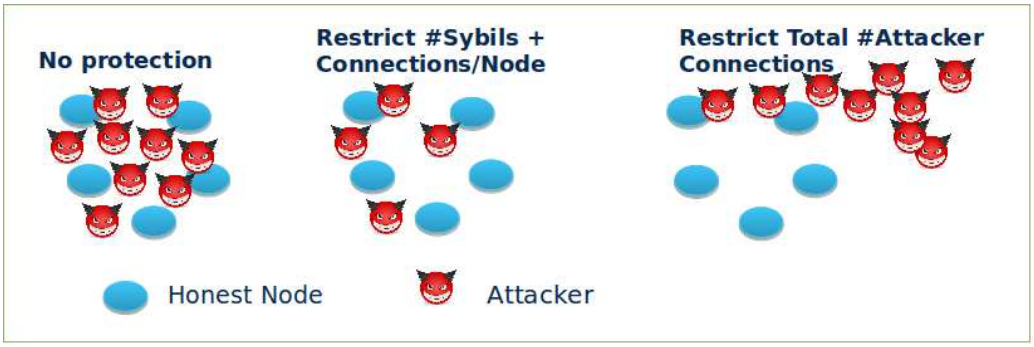


Figure 2.5: Illustrating the connectivity of attackers i) without protection schemes, ii) when restricting the number of attacking nodes, and iii) when restricting the total number of connections an attacker can establish but not necessarily the number of attacking nodes (e.g., F2F overlays)

In contrast, F2F overlays aim to limit the overall number of connections attackers can establish, thus following the second approach for attack mitigation. We assume trust relationships are costly to establish because humans only trust a limited number of people [81]. Thus, F2F overlay restrict connectivity to mutually trusted parties. In this manner, attackers can establish an arbitrary number of identities but connections to other nodes are not easy to establish. Hence, the attacker's impact is limited without requiring additional protection schemes. Furthermore, nodes are not required to connect to untrusted ever-changing strangers. F2F overlays offer a natural protection against attacks and have the added benefit of membership-concealment.

Note that all considered approaches up to now aim to reduce the probability of an attacker being involved in the communication. A complementary approach is reducing the probability that the adversary in fact executes an attack. For instance, reputation systems aim to mitigate attacks by only allowing "well-behaving" nodes to gain influence. A summary of reputation systems in P2P overlays is given in [102]. In a nutshell, reputation systems assign a score to each node. Positive behavior, e.g., supplying the required service, increases the score, while negative behavior, e.g., not responding to a request, decreases the score. Participants then preferably select neighbors with a high score, such that an attacker can only maintain a high impact if it performs adequately. In this manner, denial-of-service attacks and thus censorship is impeded. However, passive surveillance of the communication and identification of the communicating parties is not prevented. Thus, reputation systems may prevent censorship but fail to prevent de-anonymization. As a result, they are not a suitable solution on their own but could be applied for providing additional attack resistance.

2.4.2 Concepts

While F2F overlays are primarily defined by their restriction of connectivity to mutually trusted nodes, they exhibit additional common concepts.

For instance, communication between F2F overlay neighbors is *end-to-end encrypted*. Before establishing a connection, nodes exchange key material using out-of-band methods. Furthermore, the nodes exchange contact information such as IP addresses. Afterwards, they establish a connection within the F2F overlay based on the previously exchanged credentials. In this manner, the content of messages cannot be inferred by passive observers such as ISPs.

In a F2F overlay, nodes forward their requests only to trusted nodes. Nodes on the route keep track of the requests they forward based on their unique request ID. Thus, a reply can be returned along the same path. Such *hop-by-hop anonymization* provides a natural obfuscation of the communicating parties as sender and receiver do not communicate directly. So, the nodes on the path act as mixes, which obfuscate the sender and receiver against local adversaries. Ideally, nodes on the path should not be able to tell if their predecessor and successor are simply forwarding the messages or are indeed the sender or receiver.

In the remainder of this thesis, we assume that both end-to-end encryption and hop-by-hop anonymization are applied.

2.4.3 Attacks on F2F overlays

In Section 2.3.3, we categorized attack strategies for P2P overlays. We identified Sybil and Routing Table Poisoning attacks as particular threats. While F2F overlays provide a natural protection against these two attacks, it remains to determine their vulnerability to further attacks. Even if F2F overlays are not vulnerable to the current form of an attack, they might suffer from a modified attack based on the same

paradigm. In the following, we analyze the susceptibility of F2F overlays to the attacks introduced in Section 2.3 on an abstract level.

Eclipse/Black Hole attack: The main component of a Black Hole lies in the Routing Table Poisoning attack, which encourages nodes to choose the attacker as an overlay neighbor. F2F overlays thus mitigate the attack with regard to the connection establishment. However, a successful Black Hole attack also requires the honest node to forward requests to the attacker rather than its honest neighbors [108]. Manipulating distributed algorithms such that connections to attackers are preferred remains a valid attack strategy for F2F overlays. Assuming that an attacker can establish some connections to honest nodes, it might be able to advertise these connections such that it receives a large fraction of requests despite its low connectivity. If and to what extent the attacker is able to launch such an attack is highly dependent on the communication protocols of the F2F overlay. As a consequence, protocols for F2F overlays need to consider this modified Black Hole attack and protect against it.

Localized Eclipse attack: Localized Eclipse attacks aim to censor only a specific service by either i) infiltrating the neighborhood of the service providers or ii) ensuring that the service is only provided by attackers. Infiltrating a node's neighborhood is assumed to be hard in a F2F overlay, so that approach i) is not applicable in its current form. Similar to Eclipse attacks, a modified form of the attack is to increase the probability to be on a path to the respective node. Again, algorithms are required to prevent localized Eclipse attacks by design. However, the existing solutions such as randomizing the forwarding of requests [72] can be applied. Similar, case ii) remains a problem of similar impact as in an open P2P overlay and thus is likely to be resolved by similar means such as [46]. Hence, we exclude localized Eclipse attacks for this thesis due to i) the existence of likely-to-be-applicable solutions and ii) the lower impact of such attacks in contrast to global Black Hole attacks.

Pollution/Index Poisoning: Pollution and Index Poisoning remain a problem in F2F overlays. However, there exist various protection schemes [95, 96], which can be applied in F2F overlays. Indeed, the difficulty of joining the network using a new identity and thus 'whitewashing' a node should increase the effectiveness of reputation schemes. We thus disregard Pollution and Index Poisoning in this thesis in order to focus on the attacks requiring special solutions for F2F overlay.

De-anonymization: Despite the use of hop-by-hop anonymization, local internal attackers on a route might be able to identify the sender or receiver of the request. The identification of communicating parties might rely on i) timing analysis, i.e., when requests are forwarded, and ii) the content, in particular the receiver information, of a message and iii) the forwarding decisions of the nodes.

Traffic analysis has been considered in prior work both in general and for F2F overlays in particular. In the area of F2F overlays, [123] presents an in-depth analysis including adequate protection schemes for i) traffic analysis with regard to local adversaries. Global adversaries can still observe all communication and thus determine the receiver as being a node not forwarding a request. Thus, such powerful adversaries require additional protection schemes. For example, Chaum mixes in their original form guarantee anonymity against a global passive adversary observing the communication along all edges and controlling some of the nodes as long as at least one node on the path is not compromised [43]. An alternative solution is the use of cover traffic as in the anonymous P2P overlay Tarzan [68], i.e., nodes sent additional fake messages to prevent the attacker from identifying the sender and receiver as the first and last node, respectively, on a path. Thus, during this thesis, we assume that adequate protection mechanisms against traffic analysis are provided. Under this assumption, we hence focus on de-anonymization due to ii) the content of the message, and iii) the forwarding decisions attackers can observe.

We have motivated F2F overlays and evaluated their vulnerability to common attacks on an abstract level. In our evaluation, we identified (modified) Black Hole attacks and de-anonymization through the use of receiver addresses as the prevalent unsolved problems. In the light of this informal threat discussion, we now present a formal attacker model.

.

2.5 Adversary Model and Assumptions

F2F overlays aim to provide privacy-preserving communication in possibly malicious environments. For designing adequate protection measures, we first require an adversary model. The adversary model includes i) the goals of an adversary, ii) the knowledge accessible to the adversary, and iii) the capabilities of the adversary with regard to its interaction with the system.

2.5.1 Adversary Goals

We consider three adversary goals in this thesis:

1. Censorship of communication: The attacker executes a denial-of-service attack in order to prevent users from communicating with others or accessing content. We focus on global censorship rather than censorship of individual nodes or contents. For this purpose, the attacker first inserts *adversarial nodes* in the F2F overlay and establishes connections to honest nodes. Afterwards, the adversarial nodes manipulate the algorithms in order to disrupt communication. As discussed in Section 2.4, the main challenge in designing attack-resistant F2F overlays lies in preventing a modified Black Hole attack. More precisely, the attacker aims to manipulate the protocol such that a large fraction of requests does not reach the intended receiver because they are forwarded to an adversarial node.

2. Identifying communicating parties: In a F2F overlay, requests are relayed from a sender to a receiver via a path of trusted links. When observing a message, the attacker aims to identify the sender and receiver of the request. For this purpose, it can either use the content of the forwarded messages or the information about how and when the message is forwarded in the network. As discussed in Section 2.4, we focus on the attackers that aim to identify the receiver from a message's content, in particular potentially contained receiver addresses, and from the algorithm-specific forwarding decisions but disregard timing analysis.

3. Identifying participants: In contrast to most P2P overlays, F2F overlays restrict connectivity to mutually trusted nodes. Per default, they thus provide a natural membership-concealment against untrusted participants, though not against passive observers such as the ISP provider. However, providing membership-concealment requires that the topology of the overlay, corresponding to the social graph, remains hidden. Otherwise, if the attacker can infer sufficient information about the topology, even with pseudonymous node identifiers, nodes can be identified by using external information [115]. So, the attacker's third goal is to identify participants, apart from those who established a connection to it, by abusing the information provided by the F2F overlay's protocols.

All in all, we hence consider attacks on i) availability, ii) anonymity, and iii) membership-concealment of the F2F overlay.

2.5.2 Knowledge

We focus on local attackers, i.e., we restrict the knowledge of the attacker to the observations of its inserted adversarial nodes. More precisely, the attacker can observe the behavior of its overlay neighbors and can infer information about the remaining overlay from messages its adversarial nodes receive and send. A global passive attacker is disregarded for the time being on the basis that steganographic techniques can be applied to hide the F2F overlay traffic as suggested in e.g., [113].

Furthermore, we assume that each node has at least one honest neighbor. Otherwise, all communication of a node can be observed and censored, leaving the node no possibility to communicate with honest nodes.

2.5.3 Capabilities

As for the attacker's capacities, we assume an active, internal, possibly colluding attacker, who is able to drop and manipulate messages it receives. The adversary has the resources to insert an arbitrary number of colluding adversarial nodes in the network. For the time being, we assume that the attacker cannot sabotage honest nodes, e.g., by corrupted software. Rather, we focus on the insertion of fake identities, a prevalent problem in existing P2P overlays [162].

Whereas we allow the creation of an arbitrary number of *Sybils*, we bound the number of connections that an adversary can establish with honest nodes. Note that gaining connections and hence influence in a F2F overlay requires establishing real-world trust relationships. Such *social engineering attacks* are considered to be costly and difficult because they require long-term interaction between a human adversary and an honest participant. Thus, the number of connections between honest nodes and forged participants is assumed to be small. More precisely, we vary the number of connections an adversary can establish to be bound by $\mathcal{O}(\sqrt{n})$ for n honest nodes, in agreement with the assumptions in the related work [166].

In addition to restricting the adversaries capabilities with regard to its impact on the overlay topology, we restrict the computational power of the adversary to be bound by polynomial time algorithms. In particular, computationally secure cryptographic primitives can only be broken with negligible probability.

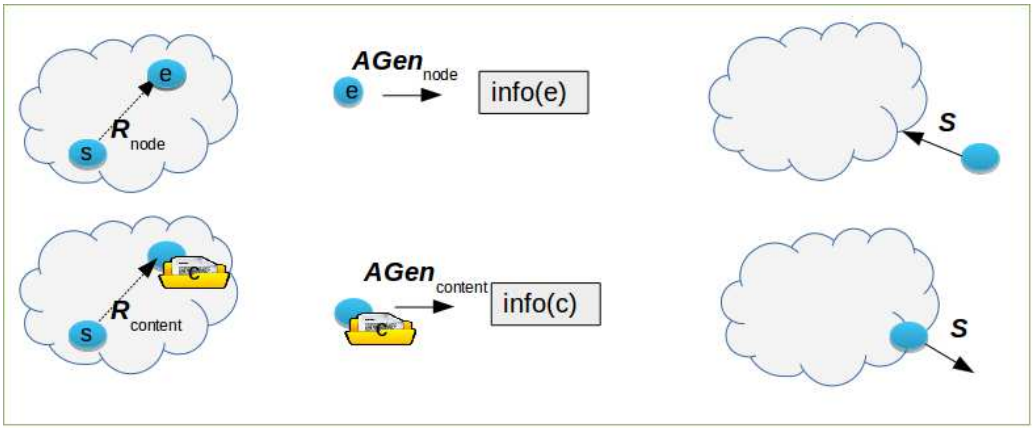


Figure 2.6: Algorithms necessary to provide basic functionalities: $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$ for discovering nodes or content, $\mathbf{AdGen}_{node}$ and $\mathbf{AdGen}_{content}$ for generating addresses for nodes or files, and $\mathbf{S}$ for stabilization in the presence of node joins and departures

In summary, we are concerned with a local, internal, active adversary aiming to undermine either the availability or the anonymity of the communication. In addition to our assumptions regarding the attacker, we assume the social graph to be of a logarithmic diameter. Such an assumption is necessary to bound the communication overhead, because the shortest paths between nodes present a lower bound on the number of messages required for communication between these nodes.

2.6 Requirements

In this section, we specify our requirements, divided into the three general aspects: *efficiency and scalability*, *robustness and censorship-resistance*, and *anonymity and membership-concealment*. We start by formalizing the functionalities of a F2F overlay before detailing the requirements of those three aspects. Complementary to our loose statement of the requirements in Chapter 1, we now provide a concise definition of the requirements, guided by our analysis of F2F overlays and our adversary model.

2.6.1 Functionalities

Recall from Section 1 that, on an abstract level, F2F overlays should provide two functionalities:

1. Messaging: Node s sends a request to node e.

2. Content Sharing: Node s sends a request to a node responsible for storing content c with the intent of either publishing or retrieving c.

Note that messaging requires s to have some possibility to address the actual receiver e, whereas content sharing only requires information about the content.

We now specify the algorithms for realizing the above functionalities. First, we need a *routing algorithm* $\mathbf{R}_{node}(s, info(e))$ for delivering a request from s to e based on an *address* $info(e)$ about e. Analogously, we need a routing algorithm $\mathbf{R}_{content}(s, info(c))$ to deliver a request concerning content c.

Therefore, the node s requires the information $info(e)$ and $info(c)$ in order to send requests. Thus, we require algorithms for i) e to generate $info(e)$ as well as $info(c)$ given c and ii) s to obtain $info(e)$ and $info(c)$. There are multiple application-specific possibilities to obtain such information, for example distributed indexes for obtaining the keys of files as in Freenet [7] or the publication of author addresses on a blog. As a result, we do not specify how exactly the information is obtained in order to remain application-independent. Rather, we focus on the algorithms $\mathbf{AdGen}_{node}$ and $\mathbf{AdGen}_{content}$ for generating $info(e)$ and $info(c)$, respectively.

Last, in order to provide a service in a dynamic system, the F2F overlay has to react to changes in its population, i.e., deal with joining and departing nodes, as well as set-up the initial system. For this purpose, we require a stabilization algorithm $\mathbf{S}$. In summary, our F2F overlay relies on 5 algorithms, namely the routing algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$, the address generation algorithm $\mathbf{AdGen}_{node}$ and $\mathbf{AdGen}_{content}$, and the stabilization algorithm $\mathbf{S}$. We illustrate the respective purpose of the algorithms in Figure 2.6.

In the following, we define the requirements in terms of the above algorithms. Our main focus in this thesis lies on improving the efficiency and scalability of F2F overlays due to their long delays and low success ratio [160].

2.6.2 Efficiency and Scalability

Efficiency implies that all algorithms have a low computation, storage, and communication overhead. Scalability is similar, indicating that the overhead increases slowly with the number of participants.

We express our bounds in terms of the number of participants n. Note that the communication complexity is generally the dominating factor in distributed systems. Local computation and storage costs when forwarding requests are generally negligible, as long as they remain polynomial in the input size. In particular, the algorithms $\mathbf{AdGen}_{node}$ and $\mathbf{AdGen}_{content}$ for generating address information of files and nodes, respectively, are local algorithms executed by a single node. Thus, we only require their computation complexity to be bound polynomial in n, the size of c, and potentially additional system-wide constants. In the following, we thus disregard local computation and storage for *forwarding* requests but consider the cost of content storage.

The routing algorithm $\mathbf{R}_{node}$ should require at most $\mathcal{O}(\log n)$ messages on average to discover a route between a randomly chosen source-destination pair. In other words, as we assume the shortest path to scale logarithmically with the number of nodes, the discovered routes should be asymptotically optimal. Such algorithms exist, as demonstrated in Section 2.2 for the case of network embeddings. While these algorithms are not directly applicable due to their lack of anonymization among others, their existence indicates the feasability of *efficient routing*, i.e., we require the algorithm $\mathbf{R}_{node}$ to have communication complexity $\mathcal{O}(\log n)$.

In contrast to $\mathbf{R}_{node}$, $\mathbf{R}_{content}$ does not only requires a low communication complexity but also a low storage complexity in terms of the maximal fraction of content stored at a single node. When storing content within a F2F overlay, distributing the content over all nodes in a balanced manner, e.g., such that the fraction of keys mapped to a node is uniformly[1] distributed, is essential. The reason for the importance of such a *balanced content addressing* is that overloading individual nodes might force them to discard files. As a result, content can be unavailable despite the existence of storage space on other nodes. In the very least, overloading individual nodes leads to congestion and incidental increased latency for retrieving content. Thus, we require that the expected maximal fraction of content stored at a single node scales with $\mathcal{O}\left(\frac{\log n}{n}\right)$. Highly successful structured P2P overlays achieve such a bound [97], thus motivating its applicability.

While routing in greedy embeddings achieves a low communication complexity $\mathcal{O}(\log n)$, greedy embeddings are not concerned with the question of (balanced) content addressing. In contrast, there exist models of social networks that assign coordinates from a lattice (e.g., Kleinberg's model [86]), such that balanced content addressing is achieved by storing a file c on the node with the closest address to $key(c)$. However, routing algorithms require more than $\mathcal{O}(\log n)$ messages within these already simplified models [101]. Thus, achieving a communication complexity of $\mathcal{O}(\log n)$ at the same time as balanced content addressing seems unrealistic in a more dynamic and less structured real-world environment. As we need to achieve balanced content addressing, we require less strict bounds on the communication complexity of $\mathbf{R}_{content}$. So, we say $\mathbf{R}_{content}$ achieves *efficient content discovery* if its communication complexity is $\mathcal{O}(polylog(n))$.

The stabilization algorithm $\mathbf{S}$ maintains the required state information for executing the algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$. In structured P2P overlays such as Chord [154], the average communication complexity of the stabilization after one node join or departure is $\mathcal{O}(polylog(n))$. As such overlays have proven to be sufficiently scalable, we also require a bound of $\mathcal{O}(polylog(n))$ messages for *efficient stabilization* [2]. Furthermore, the initialization of the state information should require at most $\mathcal{O}(npolylog(n))$ messages, because initialization can be modeled as n subsequent joins.

Simplified, we require the communication complexity of our algorithms to scale polylog and the local computation complexity to scale polynomial with the number of nodes n.

[1], or in proportion to the node's available resources, which can be realized in a system with uniform distribution by representing a node by multiple virtual nodes, see [73]

[2]The requirement assumes that the overlay is connected. If a node join merges two previously unconnected overlays, we model the join as multiple joins of all nodes from one component in the other. Similarly, a node departure dividing an overlay in two unconnected components is modeled as the combined departures of all nodes in one component and the subsequent set-up of a new overlay containing these nodes.

		E1	E2	E3	E4	R1	R2	P1	P2
Efficiency	Efficient Routing (E1)				■	■	■	■	■
	Balanced Content Addressing (E2)			■	■		■	■	■
	Efficient Content Discovery (E3)		■		■	■	■	■	■
	Efficient Stabilization (E4)	■		■		■	■	■	■
	Robustness (R1)	■		■	■			■	■
	Censorship-Resistance (R2)	■	■	■	■			■	■
	Anonymity (P1)	■	■	■	■	■	■		■
	Membership-concealment (P2)	■	■	■	■	■	■	■	

Table 2.1: Potentially conflicting requirements: Red indicates an encountered conflict.

2.6.3 Robustness and Censorship-Resistance

We characterize the robustness of the algorithms $\mathbf{R}_{node}$ or $\mathbf{R}_{content}$ as the fraction of successfully delivered requests despite a certain fraction of failed nodes. Analogously, we characterize their censorship-resistance by the fraction of successfully delivered requests in the presence of an attack. Here, we focus on the modified Black Hole attack, as introduced in Section 2.4.

We require our algorithms to improve the robustness and censorship-resistance in comparison to existing solutions. If possible, we prove that the expected fraction of successfully delivered requests is indeed higher or at least as high as for the related approach. We then quantify the actual improvement through a simulation study.

2.6.4 Anonymity and Membership-Concealment

As motivated in Section 2.4, we focus on achieving anonymity by preventing the identification of the sender or receiver of a request from the content of a message. Due to the purely local view of the adversary, possible attackers are restricted to sender, receiver, and forwarding nodes. Note that we aim to achieve anonymity even against trusted contacts in deference to the fact that trust is a multi-layered concept. Therefore, the fact that users trust each other not to reveal their presence in the system to a third party does not imply that they are comfortable sharing information about their communication.

In particular, the address information $info(e)$ in a message request should not compromise the anonymity of either sender or receiver. The anonymity should be provided against neighbors as well as any other participants. Nevertheless, the information has to reveal enough information to locate the receiver. We solve this conflict by requiring the algorithm $\mathbf{AdGen}_{node}$ to generated addresses that provide only *possible innocence* or *plausible deniability* [126]. In other words, the attacker can never identify the sender or receiver with probability 1, as there is always at least one other possible sender or receiver.

A F2F overlay provides membership-concealment against local adversaries without direct connections if the algorithms $\mathbf{R}_{node}$, $\mathbf{R}_{content}$, $\mathbf{AdGen}_{node}$, $\mathbf{AdGen}_{content}$, and $\mathbf{S}$ do not reveal identifying information about participants. However, it is generally not possible to prove that identifying information cannot be inferred. Thus, we merely argue that the revealed topology information is insufficient for breaking the membership-concealment.

This concludes the explanation of our requirements. Achieving these requirements simultaneously is challenging as they are partially conflicting. For instance, providing additional topology information usually improves the efficiency but might invite membership-concealment and de-anonymization. Similarly, increasing the stabilization complexity, e.g., by checking the presence of neighbors more frequently, usually improves the robustness as well as the efficiency of algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$. Thus, we do not only require a solution that satisfies all of the above requirements but also require the solution to offer parameters to balance between the requirements depending on the use case.

2.7 Discussion

In this chapter, we have introduced the necessary background and afterwards defined our requirements on the basis of the introduced concepts. In particular, we have provided some insights in the area of greedy embeddings, which allow locating a path between a source and a destination node in a fixed topology at a low cost. While greedy embeddings in their current form are not applicable for F2F overlays, the result motivates our requirement of efficient communication between nodes in F2F overlays. Afterwards, we have given an overview of P2P overlays and their vulnerabilities to attacks. Finding that

P2P overlays are frequently susceptible to Sybil and Routing Table Poisoning Attacks, we have identified F2F overlays as offering a natural protection against these attacks. In contrast to other solutions, they do not require additional protection schemes against the two attacks and have the added benefit of providing membership-concealment against untrusted participants. However, our evaluation of the F2F overlay concept has revealed the need for careful protocol design, as many design choices might invite censorship or de-anonymization of communicating parties. In the light of this qualitative evaluation, we have defined an appropriate adversary model and requirements for F2F overlays.

In Section 2.6, we have defined a total of eight main requirements. Four of these requirements are concerned with the efficiency and scalability of F2F overlays, namely efficient routing, balanced content addressing, efficient content discovery, and efficient stabilization. These requirements ensure a high quality-of-service, characterized by low delays for receiving the desired service and a low overhead for providing the service. The remaining four requirements are concerned with resilience, namely robustness, censorship-resilience, anonymity, and membership-concealment. These requirements ensure the provision of the service despite failures and attacks. Table 2.1 lists all requirements in addition to potential conflicts between requirements we observed during this thesis.

As indicated by the large number of potential conflicts in Table 2.1, designing an F2F overlay fulfilling all requirements is a challenging task. A particular problem is that an approach can be potentially fulfill one requirement for one set of parameters and a second requirement for a second set of parameter but cannot fulfill both requirements simultaneously. Thus, we often prove that an approach cannot fulfill our requirements by showing that two or more requirements can not be fulfilled simultaneously. In the next chapter, we start our evaluation of F2F overlays by reviewing state-of-the-art approaches.

Chapter 3

Assessing the State-of-the-Art

In this chapter, we discuss the state-of-the-art in the area of F2F overlays. In particular, we focus on the routing algorithms $\mathbf{R}_{node}$ or $\mathbf{R}_{content}$ for locating nodes or content. Therefore, we categorize the different approaches into three classes based on their routing scheme:

1. Unstructured overlays: In unstructured approaches, nodes forward requests independently of the intended receiver or the requested content.

2. Embedding-based overlays: Network embeddings assign coordinates to nodes such that the structure of the graph is reflected by the coordinates. Routing is then based upon the coordinates of the direct neighbors.

3. DHT-like overlays: DHT-like approaches aim to modify the concept of DHTs such that it can be applied in F2F overlays.

In contrast to Section 2.2, we here discuss how network embeddings have been applied in the area of F2F overlays, whereas we previously introduced the underlying concepts.

Note that many of the real-world systems offer an Opennet mode in addition to the F2F mode. A user participating in the Opennet mode connects to end devices of untrusted users. We generally focus on the F2F mode in our descriptions, but we state results for the Opennet mode if they are of interest for the F2F mode as well.

In each section, we describe several individual approaches, followed by an assessment of the concept in general. We discuss unstructured overlays, embeddings into low-dimensional l^p spaces, and DHT-like overlays in Sections 3.1, 3.2, and 3.3, respectively. In addition to defining the overlay topology in F2F networks, social graphs are applied to facilitate e.g., Sybil detection or anonymization in P2P overlays, which are not necessarily F2F overlays. As insights of these approaches, in particular their assumptions with regard to the connectivity of Sybils and honest nodes in a social graph, are of interest for our design, we give a short overview of selected systems in Section 3.4. Last, in Section 3.5, we discuss the identified gaps in the state-of-the-art and their impact on the remainder of this thesis.

3.1 Unstructured Approaches

In an unstructured approach, the local state maintained by nodes only consists of the contact information of neighboring online nodes. Thus, the stabilization algorithm $\mathbf{S}$ of unstructured approaches merely updates the neighbor lists, adding newly arrived neighbors and removing departed neighbors. Nodes forward requests non-directional, i.e., they select neighbors for forwarding the requests independently of the request's content. In this section, we discuss two unstructured F2F overlays: i) Turtle [122] uses a *flooding-based* approach, i.e., requests are forwarded to all neighbors, whereas ii) OneSwarm [15] uses *probabilistic forwarding*, i.e., requests are forwarded to a randomly selected subset of neighbors.

All approaches are designed for file-sharing with a high *replication rate*, which corresponds to the number of nodes that store a file. In other words, when a node publishes a file, it is usually stored by many nodes. Any of these nodes can return the file, so that is sufficient to find one of them.

In contrast, messaging between nodes requires that the request reaches one specific node. If the search for this node is non-directional, the search complexity is $\Omega(n)$. We show the claimed complexity as follows: Let L be a list of nodes, so that the i-th node on the list corresponds to the i-th node receiving the request. Because the order in which nodes receive the request is independent of the request's intended receiver, the expected position of the receiver on that list is $n/2$, i.e., the number of sent messages is at least $\Omega(n)$.

3.1.1 Turtle

Turtle [122] is a file-sharing system designed with the goal of anonymous file-sharing in combination with membership-concealment towards untrusted users. Turtle uses *flooding* with a limited depth to discover content, i.e., requests are forwarded to all nodes within a hop distance of at most h for a overlay-wide constant h. Files are stored by the publishers themselves and replicated on nodes that have previously retrieved them. In contrast to most other approaches, files are not indexed by keys. Rather, content can be identified by natural language descriptions.

When initiating a request, a node u includes information about the requested content as well as the intended search duration. Each request contains a query expression that describes the desired file in terms of a logical expression. In addition, the request is identified by a *queryID*, consisting of the SHA-1 of the query expression and 64 random bits, and a hops-to-live counter *htl* indicating the number of times the request may still be forwarded. This counter is initially set to a maximal value h, so that all nodes within h hops of the sender receive the query.

The content discovery and retrieval then uses flooding for content discovery, followed by the retrieval of the file along the shortest discovered path. Upon receiving a request, a node v checks its *query table* if it has already received a message with this queryID. If not so, it stores an entry of the form *(queryID, predecessor)*. Here, the *predecessor* denotes the neighbor who forwarded the request to v. If v possesses a file fitting the description, it sends a *partial reply* including the current value of h to its predecessor indicating that it can provide the file. If h is 0, v sets a flag, called the *final bit* in the reply to tell its predecessor that it will not provide any further replies. Otherwise, if h is not 0 and v has not processed the request before, v forwards the request to all neighbors (with exception of the predecessor) after decreasing the counter by 1. v forwards any replies from neighbors to its predecessor. After the initiator u has received replies from all its neighbors, u decides on the retrieval path, usually selecting a shortest path.

Turtle is unable to fulfill our requirements with regard to efficiency and scalability of the routing algorithm $\mathbf{R}_{node}$ because the communication complexity of successful flooding is at least $\Omega(n)$, as reasoned above. Depending on the choice of the maximal hop count h, nodes are either unlikely to be found or the search complexity is exhaustive. For file-sharing, the success probability is higher and the communication complexity is lower due to the replication.

In addition to the lacking scalability, the existence of the hop count allows the straight-forward identification of the requester by its neighbors and potentially reveals information about the responder. Thus, sender and receiver anonymity cannot be guaranteed, at least not towards trusted contacts.

In summary, the routing mechanism of Turtle is incompatible with our design goals.

3.1.2 OneSwarm

OneSwarm [15] is a protocol integrated within BitTorrent [1], probably the most widely used file-sharing system with up to 27 million nodes [161]. OneSwarm offers an Opennet and F2F mode following the concept of *flexible privacy*, so that each user can decide if and when it wants to connect to end devices of untrusted users. Similarly, nodes can advertise their files publicly or share them only with a selected group of friends.

As in Turtle, files are stored at the original owner as well as by nodes who have previously requested them. In contrast to Turtle, files are addressed by a key corresponding to the SHA-1 hash of their name concatenated with their content. A request then contains the low 64 bits of the key as a *searchID*. In particular, messages do not contain any information related to sender and receiver, so that the attacker can only perform timing analysis and inference from forwarding decisions in order to identify communicating parties.

Content discovery combines flooding and probabilistic forwarding. So, a forwarding node v floods the query to all trusted neighbors but forwards to an untrusted neighbor only with probability p. By only forwarding probabilistically, the attackers in the neighborhood are not able to tell that v replied to the request from the lack of a forwarded request.

When an owner of a desired file receives a request, it returns a reply informing the requester of the successful discovery. After the requesting node v has received one or several replies, v retrieves the file by either contacting the owner directly if the privacy settings allow such a connection or by requesting the file to be forwarded along the previous search path. In order to prevent queries from propagating endlessly, nodes delay requests by 150 to 300 ms. So, after successful file discovery, the requester sends search cancellation messages, which are forwarded without any delays and inform the nodes to terminate forwarding the request. In addition, the delays are supposed to prevent timing analysis and protect the responder's anonymity. In this manner, the duration and costs of the search are dynamically determined by the number of replicas of a file rather than being constant for widely replicated popular files and rare

unpopular files alike.

Whereas OneSwarm aims to reduce the costs and improve the anonymity in comparison to Turtle, it is similarly unsuitable for node discovery. Because probabilistic forwarding is also a non-directional search, the search complexity is again $\Omega(n)$. In addition, several attacks on the anonymity are possible, as detailed in [123]. So, OneSwarm's design is not suitable for our purposes, through it achieves a high performance for typical file-sharing and has a huge user base of supposedly up to 600,000 users [19].

3.1.3 Summary

In this section, we have introduced two solutions for unstructured overlays, namely flooding and probabilistic forwarding. Because they do not need to reveal any topology information, unstructured overlays achieve membership-concealment. Furthermore, there is no need for receiver addresses in the messages, such that sender and receiver anonymity is achieved in the absence of htl counters (and under the assumption that the overlay provides sufficient protection against timing analysis).

Because of the high degree of parallelism, i.e., forwarding requests along multiple paths, unstructured overlays have a high robustness and censorship-resistance against lost or dropped messages.

However, as shown in the beginning of the section, non-directional forwarding of requests results in inefficient routing. Hence, unstructured overlays are inherently unable to satisfy our requirements.

3.2 Embeddings in Low-dimensional l^p Spaces

In the context of F2F overlays, various iterative, local embedding algorithms are known to assign coordinates from low-dimensional real-valued spaces to nodes with the goal of facilitating routing, most notably the Freenet embedding.

After shortly introducing the notion of l^p spaces, we first describe Freenet, the first and only deployed embedding-based F2F overlay. Afterwards, we discuss alternative embedding and routing algorithms suggested in the context of Freenet. As none of the discussed approaches achieve a suitable performance, we review the theoretical background on embedding graphs into low-dimensional l^p spaces in order to judge if such embeddings are inherently unable to achieve efficient routing.

Note that this section differs from Sections 3.1 and 3.3 in terms of structure. The reasons of this divergence lies in the nature of the offered approaches: whereas Sections 3.1 and 3.3 describe independent solutions, Freenet's design is the prevalent solution in the area of network embeddings for F2F overlays. The remaining solutions introduced in this section only improve upon individual aspects of the Freenet algorithms.

3.2.1 Low-dimensional l^p Spaces

In this context, we somehow loosely define low-dimensional l^p spaces as normed vector spaces defined over a field $\mathbb{F}$ with norm $|,|$. The norm $||.||_p$ of a vector $x = (x_1, \ldots, x_m)$ is

$$||x||_p = \begin{cases} \left(\sum_{i=1}^{m} |x_i|^p\right)^{1/p}, & 1 \leq p < \infty \\ \max_{i=1\ldots m} |x_i|, & p = \infty \end{cases}.$$

Consequently, the distance of two vectors x and y is given by $||x - y||_p$.

For most of this section, we deal with concrete l^p spaces. For example, Freenet assigns coordinate in the unit ring $\mathbb{S}_1$, which corresponds to the interval $[0, 1)$ equipped with the distance $||x - y||_1 = \min\{|x-y|, 1-|x-y|\}$. The theoretical results mainly rely on lattices, i.e., coordinate spaces in the form $\mathbb{Z}_k^m$. However, we consider results for general l^p spaces in the last part of the section, thus motivating the very general setting.

3.2.2 Freenet

Freenet was originally advertised as an overlay for censorship-resistant publication [47, 49], which initially only offered a Opennet mode. In 2007, Freenet has been extended to include a membership-concealing F2F mode [48], based upon an iterative network embedding. Furthermore, the functionalities of Freenet have been extended beyond simple publication of content, including anonymous webpages within Freenet, anonymous email, chat, and social networking [6]. All of these components use the same application-independent algorithms and protocols for storing, finding, and retrieving content, which we discuss in the following. First, we describe the nature of node and file identifiers in Freenet. The most important aspects of the protocol with regard to our research questions are the embedding algorithm for the F2F

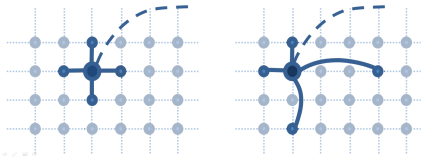


Figure 3.1: Kleinberg model [86] (left) with full neighborhood connectivity and one long range link (dashed line), opposed to our topology model [136] with connectivity within C-neighborhood and a long range link (right, further long range links omitted)

overlay and the corresponding routing algorithm. Thus, we focus on these two aspects in more detail. Afterwards, we evaluate Freenet with regard to our requirements.

Content Addressing: In Freenet, users and files are identified and verified using cryptographic keys. A user's public and private key are created upon initialization of its node and used to sign published files. In addition, each node is addressed by a coordinate, called *location*, from the unit sphere $\mathbb{S}_1$. Similar to a DHT, Freenet nodes are responsible for storing files whose key is close to their location. Freenet allows the assignment of various types of (file) keys, all sharing the same key space $\mathbb{S}_1$, as detailed in [49].

Embedding Algorithm: Both the neighbor selection in the Opennet and the F2F embedding are motivated by Kleinberg's small world model [86], which offers a potential explanation on decentralized route discovery in social networks. Kleinberg's model considers n^m nodes placed in a m-dimensional lattice equipped with the 1-norm, i.e., a l^1 space. Edges exist between all nodes at distance up to k_{short} for some $k_{short} \geq 1$. Furthermore, each node u has k_{long} *long-range neighbors* v whose distance D to v follows the distribution

$$P(D = d) \approx \frac{1}{d^m \log n}. \tag{3.1}$$

We show an example for $m = 2$ dimensions in Figure 3.1. If Equation 3.1 holds, greedy routing terminates successfully using $\mathcal{O}(\log^2 n)$ messages [86]. Hence, Freenet nodes aim to choose their neighbors or coordinates such that Equation 3.1 holds.

In the Opennet, nodes can select their neighbors such that Equation 3.1 holds. However, our measurement study in [134] revealed that the then-current neighbor selection does not achieve the desired distance distribution, resulting in recent changes to the neighbor selection algorithm. As the optimization of the Opennet is not the focus of this thesis, we defer the results of the measurement study and the proposed new algorithm to Appendix A.

In the F2F mode, nodes execute a distributed iterative embedding algorithm, which guarantees that the distance distribution of neighbors converges to Equation 3.1 in a static network. Note that Equation 3.1 states that the probability of an edge with *edge length* d decreases with d. As a result, nodes in the Freenet F2F mode aim to minimize the product of edge lengths in the overlay. In other words, the embedding algorithm aims to solve the problem

$$\text{Find } id\colon V \to C, \prod_{(u,v)\in E} ||id(v) - id(u)||_1 \text{minimal}$$

for graph G and a set of initial randomly selected of $n = |V|$ coordinates $C \subset \mathbb{S}_1$ that remains fixed during the algorithm. Freenet applies a distributed Markov Chain Monte Carlo algorithm to solve this optimization problem. During the optimization, a node u adapts its coordinate by periodically sending *swapping requests*. The swapping request is forwarded using a random walk of length 10, terminating at a approximately uniformly selected *swapping partner* v. The two nodes u and v compute the product of the distances to their neighbors, i.e.,

$$\alpha = \prod_{w\in N(v)} ||id(w) - id(v)||_1 \prod_{w\in N(u)} ||id(w) - id(u)||_1.$$

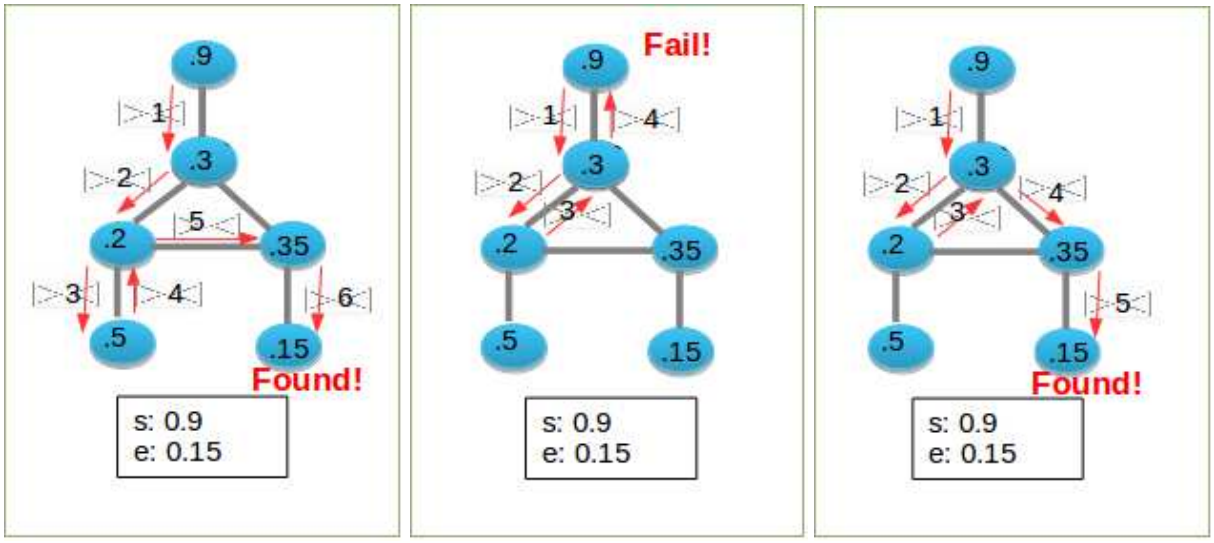
u and v swap their coordinates if α is decreased. Otherwise, if α is increased by the swap, u and v swap with a certain probability, defined by the ratio of α before and after the potential swap in order to overcome local optima of the optimization. In this manner, the distance distribution is guaranteed to eventually converge towards Equation 3.1 in a static overlay, i.e., without topology changes and attacks. A detailed analysis of the optimization algorithm is presented in [142]. In the following, we focus on the security aspects of the swapping algorithm and its inability to establish a perfect ring structure.

Routing Algorithm: Note that the swapping algorithm aims to minimize the distances between neighbors but does not guarantee that nodes closest in distance are indeed neighbors. In other words, a node might not know its successor and predecessor in the ring. Hence, the embedding is not greedy and standard greedy routing inherently fails due to local optima of the distance function. For this reason, Freenet utilizes a distance-directed depth-first search rather than the standard greedy routing algorithm.

A distance-directed depth-first search traverses the nodes in a graph in a depth-first search, so that the order in which neighbors are traversed depends on their distance. In the Freenet implementation, a node initiates a request consisting of the destination (node or file) location and a random request identifier *ID* in order to keep track of already processed requests. For each *ID*, a node u maintain the information about its predecessor, i.e., the first neighbor that forwarded the request to u, and the set of neighbors w that u forwarded the request to. If a node u receives a request from a neighbor w, u first checks if it is the destination. Otherwise, u determines if it already processed the request before. If not, u generates a record for the request *ID* and forwards the request to the neighbor with the closest location to the destination, disregarding w. Otherwise, u checks if the request is currently in the backtracking phase, i.e., if u has previously forwarded the request to w. If the request is indeed in the backtracking phase, u has to select the next neighbor for the distance-directed depth-first search by either i) forwarding the request to the closest neighbor to the destination that u has not previously forwarded the request to or ii) forwarding the request to the predecessor if u has forwarded the request to all its remaining neighbors. Otherwise, if u has not previously forwarded the request to w, u sends a message to w indicating the existence of a loop. w then considers its remaining neighbors as specified above. We present an example of how a request traverses from sender to receiver in Figure 3.2a.

If a request is concerned with storing a file in the overlay, the file is stored by any node on the path whose location is closer to the file key than any of its neighbors, by the last node on the path, and by any node that was online for at least 20 hours during the last two days. If the request is concerned with retrieving a file, the file is sent back along the same route. Nodes on the route selectively store the file.

As the above algorithm only terminates after the destination is found or all nodes in the system have received the request, a hops-to-live counter *htl* is added to ensure that the routing overhead is adequately limited. In each hop, *htl* is decreased by 1 if it is neither 1 nor the maximal value. Otherwise, if *htl* is either at its minimal or maximal value, the counter is decreased with a probability of 0.5 in order to provide sender and responder anonymity. In this manner, it remains unclear if a predecessor forwarding the request actually initiated the request or simply forwarded a request without decreasing *htl*. Similarly, it remains unclear if a node receiving a request with $htl = 1$ actually answers the request.



(a) Freenet (b) Backtracking (c) *NextBestOnce*

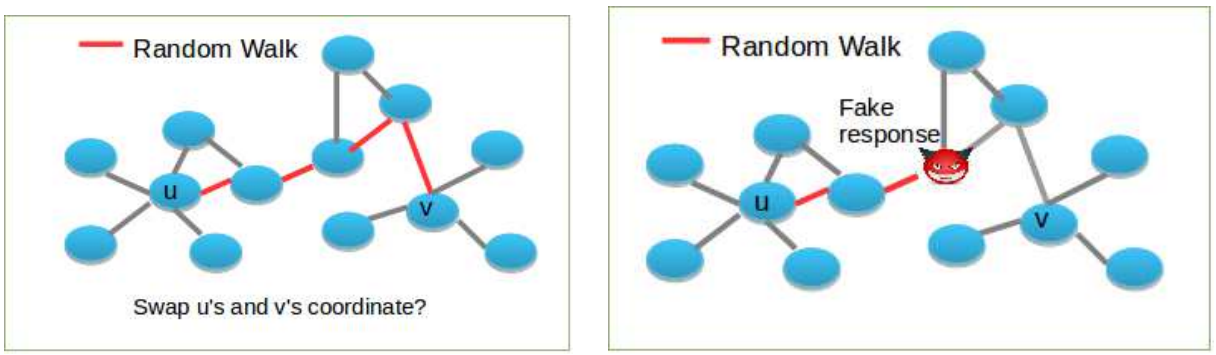
Figure 3.2: Variants of greedy routing for non-greedy embeddings: a) the Freenet routing is a distance-directed depth-first search, i.e., the order by which neighbors are selected in the depth-first search is determined by their distance to the destination. b) Backtracking traverses back along the path, possibly finding an alternative path if one of the nodes has a closer neighbor and leading to a failure otherwise. c) *NextBestOnce* allows nodes to select their predecessors a second time if none of their remaining neighbors is closer to the destination than the predecessor.

Freenet and our Requirements: We present three major concerns with regard to Freenet's suitability as a F2F overlay, concerning the efficiency of the routing algorithm, the resistance to attacks on the embedding algorithm, and the anonymity of the routing.

In previous work, we proved that Freenet's routing algorithm does not achieve polylog complexity in a non-greedy embedding. While the embedding in Freenet is motivated by Kleinberg's model, the proof in the model heavily relies on the existence of edges between closest nodes. Thus, Kleinberg's results are not valid for the modified algorithm.

In order to analyze Freenet's routing algorithm, we hence modified Kleinberg's model to incorporate non-greedy embeddings while still assuming that an embedding provides a certain local structure. In other words, we assumed that nodes share an edge with at least one neighbor within distance C. The differences between Kleinberg's model and our modification are illustrated in Figure 3.1. In the context of our model, we showed that a distance-directed depth-first search as currently implemented in Freenet cannot discover paths of a polylog length if $C > 2$, i.e., if a significant fraction of nodes do not have a neighbor that is at least the second closest node in the lattice. Furthermore, we have shown that the Freenet embedding does not provide the desired accuracy $C > 2$ [129, 136], so that the combination of the current embedding and routing scheme does not satisfy our requirements for an efficient routing scheme.

In addition to scalability and efficiency, Freenet is vulnerable to attacks. The swapping algorithm allows for denial-of-service attacks and thus censorship. Adversaries on the random walk can offer arbitrary coordinates for swapping, as illustrated in Figure 3.3. They can undermine the swapping algorithm by inserting a large number of similar coordinates in the overlay, As a result of this attack, all node coordinates *cluster* within a certain range. Such clustering results in overloading individual nodes and reduces the routing efficiency (further), as shown in [64].



(a) Swapping Requests

(b) Attack on Swapping

Figure 3.3: Freenet's embedding algorithm relies on the swapping of coordinates between endpoints of a random walk. However, an attacker can intercept swapping requests and fake responses with arbitrary coordinates.

Last, Freenet's indeterministic reduction of the *htl* counter provides anonymity for file-sharing. However, messaging with a concrete receiver coordinate is not anonymous due to the uniqueness of the coordinate, which reveals the receiver. Furthermore, Freenet gathers supposedly anonymous statistics, which can reveal identifying properties of nodes. As those statistics are not required for providing basic functionalities considered in this thesis, we refer to Appendix B for a detailed explanation. In addition, revealing the coordinates of a node and its neighbors to the swapping partner provides an adversary with the opportunity to reconstruct the topology of the social graph by combining the neighborhood information from a large number of intercepted swapping requests. Thus, the information revealed during swapping might undermine the membership-concealment.

In summary, Freenet fails to fulfill our requirements with regard to efficiency, censorship-resistance, anonymization and possibly membership-concealment. As a consequence, we now consider alternative embedding and routing algorithms suggested in the related work.

3.2.3 Alternative Embedding Algorithms

We review two alternative embedding algorithms. Both mitigate the vulnerability of the coordinate swapping by basing the iterative adaption of coordinates only on the coordinates of the direct neighbors.

LMC: In previous work, we extended Evans et al.'s work [64] on the vulnerabilities of the Freenet embedding by strategically designing various attack strategies and evaluating them in large-scale social

networks. Based on the insights of this evaluation, we then presented LMC, the Local Markov Chain embedding, which can successfully mitigate the proposed attack strategies [143]. LMC replaces the periodic selection of a swapping partner with the period selection of a random coordinate x. However, LMC results in clusters of coordinates in certain areas of the coordinate space. As a consequence, the majority of files is mapped to the small fraction of nodes not contained in these clusters, so that LMC does not offer balanced content addressing.

Dell'Amico: Dell'Amico modified a graph drawing algorithm [89], which assigns node coordinates in a m-dimensional l^p space, such that it is executed in a distributed manner using only local information [59]. Like the Freenet embedding, the algorithm is iterative. The general idea of each step is that a node moves its coordinate towards the center of mass of its neighbors' coordinates and then normalizes its coordinate to prevent clustering of coordinates. While simulations indicate that the algorithm reduces the length of the routes in comparison to the Freenet embedding, the author does not provide any results about its scalability, content addressing, and censorship-resistance. As a consequence, it remains unclear if the proposed algorithm satisfies our requirements. Anyways, it does not provide a greedy embedding and thus does not provide polylog routing overhead whenever a distance-directed depth-first search is applied. However, alternative routing algorithms might enable the discovery of short routes.

3.2.4 Alternative Routing Algorithms

We have shown that the Freenet routing is inefficient. Hence, we now review existing alternatives. Figure 3.2 exemplary compares the graph traversals of these alternatives with the original Freenet routing algorithm.

Backtracking: In [142], Sandberg evaluates a backtracking algorithm in addition to a directed depth-first search. Here, a node returned a message to its predecessor if none of its neighbors' coordinates is closer to the destination than its own. In this manner, the algorithm discovers alternative greedy paths along which the distance decreases in each step even if the initially selected path terminates in a local optimum. However, as such an algorithm requires the existence of at least one path with a monotonously decreasing distance, the success ratio of the routing is low. For instance, in Figure 3.2b, the node with coordinate 0.3 does not forward the request to the node with coordinate 0.35 because 0.35 is at a larger distance from the destination 0.15. So, backtracking is not a suitable alternative for non-greedy embeddings.

NextBestOnce: In previous work [135], we identified that a main drawback of the Freenet routing algorithm lies in forwarding requests to a neighbor u at a high distance to the destination even if a different but previously contacted neighbor v is closer to the destination. Due to its low distance to the destination, v is likely to have several neighbors that are closer to the destination than u. Thus, forwarding a request to v multiple times can be more promising than choosing a new node u.

Based on the above insight, we proposed *NextBestK* as an alternative routing algorithm for Freenet. Our proposed algorithm *NextBestK* permits each node u to forward the request multiple times until u has forwarded the request to at most K neighbors at a larger distance to the destination than u itself. In this manner, nodes close to destination can consider all their close neighbors before a request is forwarded via a long-range link. Figure 3.2c displays an example, showing that rather than forwarding a request to the distant node with coordinate 0.5, the node with coordinate 0.2 first considers its predecessor 0.3 in order to check if 0.3, being closer to the destination 0.15, can provide a route via an additional neighbor. In combination with a backtracking phase, during which distant neighbors such as 0.5 are considered as well, *NextBestK* can achieve guaranteed success.

We provided asymptotic bounds on the routing complexity of *NextBestOnce*, i.e., *NextBestK* with $K = 1$, in the context of our modified small-world model, assuming that each node has a neighbor within distance $C(n)$ in each principal direction, as illustrated in Figure 3.1. If and only if $C(n)$ is polylog, the algorithm achieves polylog complexity. However, *NextBestOnce* does not achieve logarithmic complexity.

Neighbor-of-Neighbor Routing: Now, we discuss the use of Neighbor-of-Neighbor (NoN) information for routing, i.e., basing the routing decision not only on the coordinates of neighbors but also on those of the neighbor's neighbors. In other words, nodes forwarded a request to the neighbor with the closest coordinate to the destination in its neighborhood. For instance, the current Freenet implementation optionally considers NoN information to reduce the complexity of the routing. Because the additional information complicates the realization of sender and receiver anonymity as well as membership-concealment, we generally prefer algorithms based purely on neighbor information.

While NoN information should result in less messages in concrete scenarios, it is not immediately clear if such information reduces the asymptotic complexity if the social graph exhibits a scale-free degree distribution. In previous work [129, 138], we evaluated *NextBestOnce-NoN*, a modified version of the previously introduced *NextBestOnce* that includes NoN information. We proved that, in the context of our model, such information indeed reduces the asymptotic communication complexity. Nevertheless, our upper bound on the routing complexity slightly exceeds the desired logarithmic growth. As a consequence, the asymptotic bound on the routing length of *NextBestOnce-NoN* still fails to satisfy our requirements.

3.2.5 General Bounds

In this section, we first consider the question if efficient routing with $\mathcal{O}(\log n)$ messages is possible. Afterwards, we relax the problem and question the existence of algorithms with a polylog complexity.

In order to answer the first question, we consider an idealized scenario. Like Kleinberg's model, we assume that nodes are assigned equidistant coordinates from a lattice and nodes with the closest coordinates are indeed adjacent. In other words, the local structure of the overlay is optimal as greedy routing is possible and nodes are spread equidistant to allow for balanced content addressing. Kleinberg showed that if the distance between neighbors does not follow Equation 3.1, greedy routing does not provide polylog paths. Hence, we assume that Equation 3.1 holds. Then, the lower bound on the hop count is $\Omega\left(\log^2 n\right)$ for a constant degree distribution [101] and $\Omega\left(\log^{\alpha-1} n\right)$ for a scale-free degree distribution in one dimension [66]. Hence, low-dimensional embeddings in l^p seemingly do not allow the discovery of logarithmic paths.

Now, we consider the question if at least polylog complexity is possible. If we assume an underlying lattice, i.e., the embedding is greedy with connections between all nodes at distance 1, and a distance distribution following Equation 3.1, polylog routing follows from Kleinberg's model. So, we now assume that the embedding is not greedy. In the context of our model, we showed that polylog routing in a non-greedy embedding requires the distance to the closest neighbor $C(n)$ to scale polylog [138]. Thus, we can reduce the question if embeddings into l^p spaces allow such a polylog $C(n)$ to our problem of finding polylog routes. The first question is closely related to a result about the stretch of a network embedding, i.e., the normalized upper bound on the distance between neighbors. It was shown that there exists graphs which cannot be embedded with a polylog stretch [103]. The result does not conclusively refute polylog routing in low-dimensional l^p-embeddings because i) the existence of such graphs does not necessary imply that the result holds for social networks, and more importantly ii) the existence of some neighbors at a larger distance does not prevent polylog routing as long as at least one neighbor is within polylog distance. Nevertheless, a polylog $C(n)$ requires that a connected graph with a subset of edges is embedded with a polylog stretch. Thus, the result heavily indicates that providing a polylog $C(n)$ is not possible, especially since it assumes global knowledge and arbitrary computation power, which both cannot be provided in F2F overlays. Hence, embedding in low-dimensional l^p spaces are unlikely to provide the desired polylog communication complexity.

3.2.6 Summary

We recapitulate and generalize our assessment of network embeddings into low-dimensional l^p spaces for F2F overlays in terms of our requirements.

Network embeddings adapt node coordinates to the topology, thus potentially revealing essential information about the topology to an attacker. As a consequence, it remains unclear if they indeed conceal the membership of users. The revealed information might enable correlation of node coordinates and real-world users. Furthermore, they fail to provide receiver anonymity if the node coordinates are used as receiver addresses in the messages.

Robustness and censorship-resistance of these embeddings has not been analyzed in depth. However, the use of backtracking and re-routing after time-outs allows the discovery of alternative paths in the presence of failures or intentional sabotage of the forwarding. Thus, an attacker can only increase the delays by dropping messages but does not reduce the success ratio. In addition, the attacker can sabotage the embedding algorithm. In the absence of contradicting evidence, we declare the respective embeddings censorship-resistance, i.e., we assume that LMC and Dell'Amico offer censorship-resistance. However, if we were to analyze these embeddings in grater detail, a more profound evaluation would be in order.

One of the main issues of these embeddings is their scalability. While Freenet does not provide the desired scalability, we are unable to show that lacking scalability is an inherent trait of the approach. We showed that they are inherently unable to provide efficient routing in $\mathcal{O}\left(\log n\right)$ messages. However, the state-of-the-art does not conclusively show the inability of embeddings into low-dimensional l^p spaces to provide polylog complexity but only strongly indicates the impossibility of polylog communication

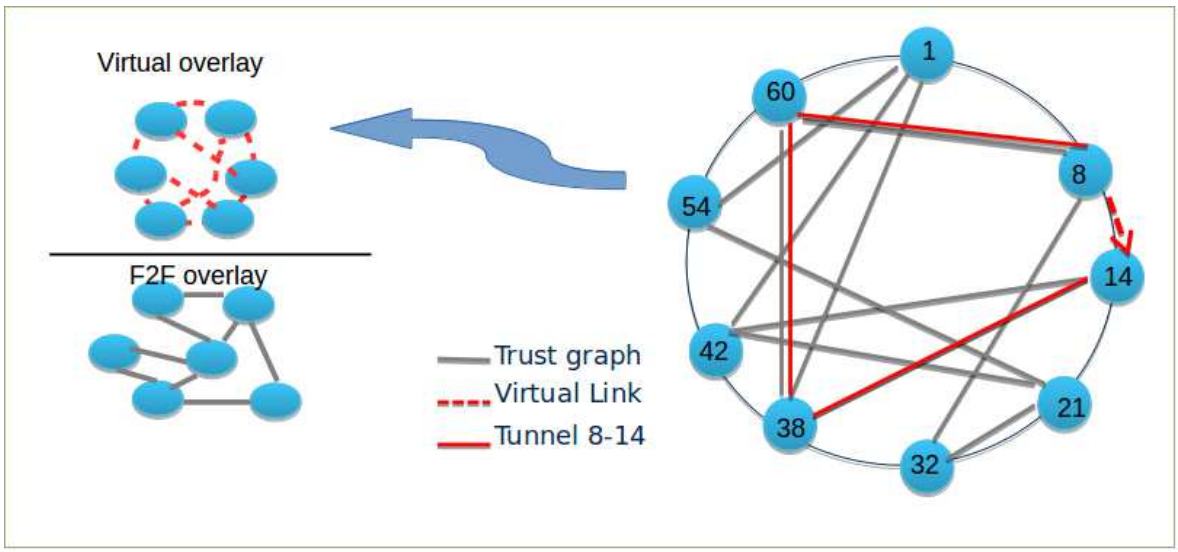


Figure 3.4: Concepts of Virtual Overlays
left: Virtual overlays are 2-level overlays establishing a structured overlay on top of the F2F overlay.
right: Example of a virtual link: rather than directly communicating, the virtual neighbors with coordinates 8 and 14 communicate via a tunnel using only trusted links.

complexity. As a consequence, we disregard low-dimensional l^p-embeddings in the remainder of this thesis but for a simulation-based comparison with our own design.

3.3 DHT-like Overlays

In this section, we introduce the concept of DHT-like overlays in the context of F2F overlays. DHTs, as introduced in Section 2.3, offer efficient routing by assigning coordinates to nodes and arranging the overlay topology such that finding the closest node coordinate to any coordinate requires $\mathcal{O}(\log n)$ messages. However, in F2F overlays, the social graph dictates the overlay topology and cannot be adjusted in order to enable efficient routing. Nevertheless, several designs aim to leverage the benefits of DHTs. We distinguish two general ideas for realizing DHTs in F2F overlays.

The first approach is to realize DHTs without establishing the necessary overlay connections to guarantee successful routing. Nodes in the F2F overlay choose random coordinates and forward requests based on the coordinates of direct neighbors in the F2F overlay. In this manner, the neighbor selection is independent of the neighbor's coordinate whereas the neighbor selection in DHTs is based predominantly upon coordinates. Thus, the neighbors in the F2F overlay do not correspond to the designated neighbors in the DHT. As a consequence, the deterministic routing algorithm is likely to fail. In order to nevertheless discover content, these approaches use a high degree of parallelism in combination with a high replication rate.

In contrast, the second approach is to establish the required structure in a virtual overlay, as illustrated in Figure 3.4. Because virtual neighbors are unable to establish connections in the F2F overlay, they communicate indirectly via a path, also called *tunnel* or *trail*, so that two subsequent nodes on the path represent users with a mutual trust relationship. However, for the design to be efficient, these tunnels have to be short as well as efficiently maintainable in the presence of network dynamics.

In this section, we start by introducing GNUnet [65], a real-world system following the first approach. We point out that GNUnet suffers from similar drawbacks as unstructured overlays. In the remainder of this section, we focus on virtual overlays, introducing the two F2F overlays X-Vine [110] and MCON [160].

3.3.1 GNUnet

GNUnet has been an active project for more than a decade [33] during which the project underwent multiple changes. Our description of the routing scheme is based upon [65].

Like OneSwarm, GNUnet offers both a F2F mode and a Opennet mode. In the F2F mode, GNUnet aims to use the Kademlia DHT [104] without changing the topology or maintaining additional information expect the neighbor list. So, as in Kademlia, each node is assigned a 160-bit coordinate with the distance of two coordinates corresponding to their XOR. Neighbors of a node v are then grouped into *buckets*

according to the common prefix length of coordinates. In order to realize efficient file-sharing, files are assigned a key corresponding to the SHA-1 hash of their content. Note that by restricting the entries in the buckets to nodes of trusted participants, the F2F overlay does not provide the topology required by the standard Kademlia routing algorithm to guarantee the discovery of keys using $\mathcal{O}(\log n)$ messages.

Thus, GNUnet modifies the routing algorithm of recursive Kademlia [78] such that nodes publish a high number of replicas and routing uses a high degree of parallelism. In this manner, the probability to successfully locate content is increased to be close to 1. In order to locate a node responsible for a certain file, the routing of *PUT* requests for storing a file or *GET* requests for retrieving a file relies on two phases. During the first phase, requests are forwarded randomly to select initiators of the second phase approximately uniformly at random from all nodes. These initiators then execute deterministic routing by forwarding the request to the neighbor closest to the request key. The deterministic routing terminates if a node can satisfy a *GET* request or does not have a neighbor closer to the destination. For a *PUT* request, all nodes closer to the file's key than any of their neighbors store the file. Each request contains a hop counter and the routing changes from the first phase to the second phase after $\log n$ hops, with n being estimated using a secure completely distributed algorithm [63].

The authors show that for a overlay with c neighbors per node, $\sqrt{\frac{n}{c+1}}$ replicas are required to locate content with overwhelming probability. Consequently, they propose an algorithm for spreading *PUT* requests in such a manner that the replicas indeed end up at approximately $\sqrt{\frac{n}{c+1}}$ nodes. Constant-degree graphs exhibit a radically different structure to social networks, so that results with regard to the number of replicas do not necessarily apply for F2F overlays. However, as social networks have been shown to exhibit a scale-free degree distribution and hence a constant average degree, the results indicate that the number of replicas scales linearly with n. Furthermore, the number of messages required for content discovery are $\mathcal{O}\left(\sqrt{n}\log n\right)$.

GNUnet fails to satisfy our requirements with regard to two aspects. First, the hop counter reveals the sender's identity to the first node on the path. Hence, GNUnet does not provide sender anonymity. Whereas the lack of sender anonymity can be fixed by probabilistically decreasing the hop counter, the second aspect is a general drawback of all approaches that restrict the stabilization to neighbor list updates. Even with the high replication rate, GNUnet cannot achieve polylog routing, thus it does not satisfy our requirements with regard to efficiency.

3.3.2 X-Vine

X-Vine [110] uses F2F overlays with the goal of improving Sybil resistance without revealing a node's trusted neighbors. Whereas membership-concealment is not a primary goal of the authors, the presented routing and stabilization algorithms do not principally require revealing the participation of users. Some key ideas of the protocol have previously been suggested in the context of sensor networks [28, 40]. However, the scenario of sensor networks is different as the considered networks are of a smaller size and attack resistance is not of interest.

X-Vine establishes a virtual ring similar to the Chord overlay on top of the F2F overlay, as exemplary illustrated in Figure 3.4. Nodes choose random coordinates and establish tunnels, which are referred to as *trails* in X-Vine, to their predecessor and successor on the ring, i.e., the node with the next lower and next higher coordinate modulo the length of the ring. Additional tunnels to nodes at a larger distance are then constructed leveraging the existing tunnels. Routing in such a virtual overlay then corresponds to forwarding a request along the tunnel whose endpoint's coordinate is closest to the requested coordinate.

In the following, we first describe the nature of tunnels and the routing of requests along these tunnels. Afterwards, we describe the construction of the overlay and its stabilization in the presence of node joins and departures. Last, we present X-Vine's protection schemes against Sybil attacks and summarize the results of the evaluation.

Tunnels between virtual overlay neighbors represent a path in F2F overlay, i.e., consecutive nodes on the path share a mutual trust relationship. All intermediate nodes on a tunnel maintain a *trail record* consisting of the start- and endpoint's coordinate as well as the contact information of the predecessor and the successor on the tunnel. Furthermore, the length of the tunnel needs to be tracked in order to identify unfavorable long tunnels.

When initializing or forwarding a request, nodes first select the *trail record* with the closest endpoint to the destination. Then, they forward the request to successor on that trail. Note that it is not necessary for a request to traverse complete tunnels. Rather, each node selects a tunnel endpoint independently of the previously selected tunnel. Essentially, the introduced routing protocol corresponds to a greedy routing algorithm, which always forwards a request to the neighbor closest to the destination. However, instead of considering the neighbor's coordinate, X-Vine routing considers the coordinates of the endpoints of tunnels containing the respective neighbor.

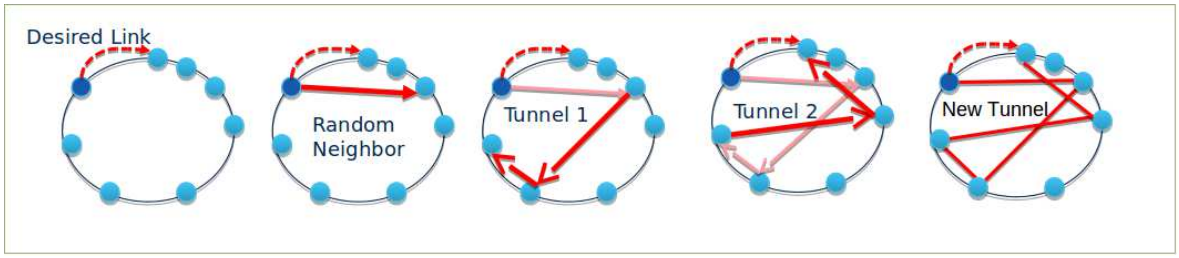


Figure 3.5: Tunnel construction to integrate the newly joined node (dark blue) by establishing a tunnel to its successor in the ring

The complexity of the routing is determined by the number of traversed tunnels and the length of the tunnels. More precisely, an upper bound on the number of required messages is given by product of the average tunnel length and the average number of tunnels in each route. Thus, under the assumption that DHT routing requires $\mathcal{O}(\log n)$ tunnels and the tunnel length is similar to the $\mathcal{O}(\log n)$ diameter, routing requires $\mathcal{O}(\log^2 n)$ messages.

Now, we explain how tunnels are constructed. Nodes joins the overlay subsequently and each node establishes its tunnels by leveraging the existing tunnels. So, new tunnels corresponds to routes in the overlay. A joining node first establishes an initial tunnel to its successor by relaying a request addressed to its own coordinate through a random F2F overlay neighbor. As this neighbor is already included in the ring, it can forward a request along the ring to the newly joined node's successor in the virtual ring. The discovered route is then used as a tunnel. We illustrate an example of the tunnel construction in Figure 3.5. This initial tunnel to its new successor in the ring is then used to establish another tunnel to the predecessor, thus reconnecting the ring through the new node. Tunnels to additional nodes can now be set up by leveraging the ring routing to send requests addressed to suitable coordinates.

When a node u departs the overlay, the remaining nodes have to re-establish tunnels with endpoint or intermediate node u. For this purpose, the neighbors of u consider all trail records for which u is their successor. For each such tunnel, they inform the startpoint of the tunnel of the failure. The startpoint then re-establishes the tunnels by sending a request for the respective endpoint coordinate.

As described above, new tunnels corresponds to routes in the overlay. Thus, they are generally a concatenation of existing tunnels. As a result, the average length of the tunnels increases over time. Thus, X-Vine extends the stabilization protocol by two additional algorithms. First, they aim to find alternative tunnels if a tunnel's length exceeds a certain threshold. Second, nodes with a high number of tunnels requests startpoints of long tunnels to reroute their tunnels. In this manner, overly long tunnels are avoided.

In addition to providing efficient communication in F2F overlays, X-Vine aims to reduce the number of Sybils. The authors suggest two protection mechanisms, both based upon the assumption that the number of edges between Sybil nodes and honest nodes is low. First, nodes restrict the number of tunnels per overlay connection, i.e., per link in the social graph. As a consequence, Sybil nodes appear only in a limited number of routing tables because only a small number of tunnels to nodes within the Sybil community exists. Second, nodes check if their virtual neighbors have established a sufficient number of tunnels by querying the supposed endpoints of their tunnels. If Sybils fail to establish enough tunnels due to the restrictions on the number of tunnels per link, honest nodes exclude those Sybils from their routing table. While the first approach restricts the number of connections the Sybil nodes can establish in total, the second approach restricts the number of Sybil identities with tunnels to honest nodes. In combination, the two approaches mitigate the connectivity of Sybils and thus their impact on the overlay.

Now, we shortly summarize the results of X-Vine's evaluation, which considers the effect of the enhanced stabilization algorithms and the Sybil defenses. The simulation-based evaluation using networks of several ten thousands nodes shows that the additional stabilization algorithms provide a decreased mean tunnel length in comparison to the basic approach. However, the system behavior over time is only analyzed by simulating sequential joins, not node departures. Departing nodes are only considered in terms of the routing success under concurrent failures. The evaluation of the Sybil defenses indicates that the fraction of honest nodes accidentally excluded due to violating the threshold is low, so that the defenses rarely reduce the quality of service for honest users.

In summary, X-Vine appears to be a very promising approach for an efficient and resilient F2F overlay. However, the limited analysis of the system's long-term behavior raises the question if the tunnel length and the stabilization complexity remain similarly low when nodes depart and rejoin the network over an extended period of time.

3.3.3 MCON

MCON [160] aims to provide membership-concealment against other participants and robustness against failures. Its design and assumptions differ from X-Vine in a number of points. Most notably, MCON relies on a centralized trusted third party for bootstrapping and enforcing restrictions on the number of neighbors. The use of a trusted third party disqualifies MCON as we require a completely distributed system. Nevertheless, we explain the overlay structure as well as the routing and stabilization algorithms of MCON. The key ideas of these algorithms are valid without the use of a trusted third party.

For the overlay topology, MCON uses a Kademlia-like overlay rather than a ring overlay. Due to Kademlia's redundant neighbor selection [104], MCON is very robust to failures.

MCON's routing algorithm is similar to X-Vine's in the sense that nodes forward requests based on the closest endpoint to the destination. However, MCON uses Kademlia's parallel routing algorithm. As the use of parallel routing decreases the impact of failures and attacks, parallel routing increases the robustness and censorship-resistance. Furthermore, routes in MCON consists of complete tunnels. In other words, intermediate nodes on the tunnel forward the request until the endpoint of a previously selected tunnel is reached. The endpoint then decides on the next tunnel. In this manner, end-to-end encryption between tunnel start- and endpoint can be applied.

However, nodes establish tunnels using flooding. We claim that the stabilization thus requires at least complexity $\Omega(n)$. Note that successful routing in Kademlia is only guaranteed if each node is connected to the node with the closest coordinate to its own [104]. As argued in Section 3.1, finding a specific node using flooding requires a complexity of $\Omega(n)$ messages. Thus, stabilization in MCON is highly inefficient.

So, despite its short routes and high robustness, MCON does not satisfy our requirements due to its exhaustive stabilization complexity and the use of a trusted third party.

3.3.4 Summary

In this section, we have seen that DHT-like overlays follow two approaches. The first approach, realized in GNUnet, is to apply DHT routing without maintaining structural information apart from neighbor coordinates. In contrast, the second approach of virtual overlays establishes indirect connections between overlay neighbors in order to allow for efficient deterministic routing.

GNUnet is unable to fulfill our requirements due to its inefficient routing. In contrast, virtual overlays, in particular X-Vine, are a promising approach. In the following, we summarize their suitability with regard to our requirements.

As argued in Section 2.2, the use of coordinates as pseudonyms can impair membership-concealment and anonymity. However, the coordinates in virtual overlays are chosen independently of the F2F overlay topology and thus the social graph. So, they prevent an attacker from reconstructing the topology based on information in the requests. However, if requests are addressed to unique nodes, neighbors can identify the receiver of the request. Thus, without obfuscating the coordinates of the destination, virtual overlays do not provide receiver anonymity against neighbors but pseudonymity.

In contrast to unstructured overlays, the resilience of the virtual overlays is generally lower, because the deterministic lookup does not consider all paths between a source and destination. Nevertheless, simulations in both X-Vine and MCON indicate that virtual overlays can achieve a high robustness and censorship-resistance.

Virtual overlays achieve a communication complexity of $\mathcal{O}\left(\log^2 n\right)$ under the assumption of short tunnels. Thus, they fail to provide efficient routing using $\mathcal{O}\left(\log n\right)$ as defined in Section 2.6. But they seem promising solutions for fast and resilient content discovery. However, it remains unclear if they can maintain short routes at an acceptable stabilization complexity.

As a consequence, we consider the question if the virtual overlay approach in general, rather than the specific realization, is able to provide efficient content discovery and stabilization simultaneously. Our results of virtual overlays, presented in Chapter 6, implicitly cover GNUnet-type systems.

3.4 F2F-aided P2P Systems

The idea of leveraging social trust to improve the privacy or censorship-resistance is not limited to the use of F2F overlays. F2F-aided P2P systems allow open connectivity between arbitrary nodes but utilize the social graph to reduce the chance of communicating with a malicious node. In this section, we introduce two applications of F2F-aided P2P systems, namely mitigation of Sybil attacks and proxy selection in an anonymity service.

3.4.1 Sybil Resistance

There are various attempts to mitigate the impact of Sybils on P2P systems, in particular SybilGuard [166], SybilLimit [165], Whanau [94], and SybilInfer [58]. All four mitigation schemes aim to reduce the number of Sybils an adversary can insert into the overlay. For this purpose, they rely upon two key assumptions:

1. The number of edges between honest nodes and adversaries is low, with the actually definition of low varying between $o\left(\frac{\sqrt{n}}{\log n}\right)$ and $\mathcal{O}\left(\frac{n}{\log n}\right)$.

2. Social networks are fast-mixing [149], i.e., the endpoint of a weighted random walk of length $\mathcal{O}(\log n)$ hops is selected approximately uniformly at random from all nodes, independent of the initiator of the random walk.

Based on the above assumptions, nodes use random walks to estimate the likelihood that certain nodes are Sybils. The key idea of the proposed algorithms is the low probability of a random walk to terminate within the Sybil community, so that only a low number of Sybils are accepted as honest nodes, namely those for which the random walks contain edges to malicious nodes. The assumed property of social networks to be fast-mixing guarantees a low false positive rate, i.e., a low probability to accidentally label an honest node as a Sybil. However, recent work indicates that social networks are not necessarily fast-mixing [112], meaning that the positive results for the detection algorithm obtained on selected social graphs might not be globally applicable. Nevertheless, the proposed algorithms offer Sybil resistance without restricting the connectivity of the P2P network. The proposed Sybil defenses present a complementary approach that can be integrated into our F2F overlay to further mitigate the impact of Sybils.

3.4.2 Proxy Selection in Anonymity Services

The P2P system Torsk [105] offers anonymization using onion routing in the manner of Tor. Nodes select a path of onion routers, randomly selected from nodes in the network, and forward an encrypted request via this path. So, only the first node can identify the sender and only the last node can decrypt the receiver's address. Hence, as long as at least one node on the path is honest, an adversary is unable to link sender and receiver. In Torsk, nodes select their routers by performing a lookup in a DHT, allowing a close to uniform selection from all nodes in the system. In this manner, the probability of selecting a malicious node for a router increases with the fraction of Sybils and thus the probability of linking sender and receiver.

Pisces [111] mitigates the impact of Sybils by selecting the onion routers based on weighted random walks within the social graph rather than DHT lookups. As for the Sybil mitigation, the defense mechanism relies upon the assumption of a low number of edges between honest and malicious nodes. Simulations on real-world social graphs indicate that indeed the probability of a successful attack is drastically reduced. Note that in contrast to the above Sybil attack mitigation algorithm, the proposed scheme does not rely upon the fast-mixing property of social networks for reducing the influence of the adversary. Only the randomness of the router selection might be reduced if the graph is not fast-mixing as nodes are more likely to select nodes in their neighborhood. By leveraging the social graph for the selection of onion routers, Pisces might also allow the integration of onion routing in F2F overlays. Like Sybil defenses, Pisces presents a complementary approach that can be integrated into our F2F overlay to increase the degree of anonymity beyond that of possible innocence.

3.4.3 Summary

We have seen that leveraging the social graph of overlay participants to increase attack resistance or anonymization is a key idea of multiple approaches, not only in the field of F2F overlays. The presented approaches might be modified and integrated in our F2F overlays complementary to our own protection schemes. However, such systems are not of interest on their own if users aim to hide their participation in the network from untrusted participants.

3.5 Discussion

In this chapter, we have introduced the state-of-the-art with regard to F2F networks. We summarize our results in Table 3.1. In addition to the state-of-the-art, we consider greedy embeddings introduced in Section 2.2, which present a interesting concept for F2F overlays but have not been considered in that area yet.

Approach		E1	E2	E3	E4	R1	R2	P1	P2
Unstructured	Turtle [122]	X	✓	X	✓	✓	✓	X	✓
	OneSwarm [15]	X	✓	X	✓	✓	✓	✓	✓
DHT-like overlays	GNUnet [65]	X	✓	X	✓	✓	✓	X	✓
	XVine [110]	X	✓	✓	?	✓	✓	X	✓
	MCON [160]	X	✓	✓	X	✓	✓	X	✓
l^p Embeddings	Freenet [48]	X	✓	X	?	✓	X	X	?
	LMC [143]	X	X	X	?	✓	✓	X	?
	Dell'Amico [59]	X	?	X(?)	?	?	?	X	?
	NextBestOnce [136]	X	✓	X(?)	?	✓	✓	X	?
Greedy Embeddings	PIE [80]	✓	?	✓	?	X	X	X	?

Table 3.1: State-of-the-art approaches with regard to our requirements: E1-Efficient Routing, E2-Balanced Content Addressing, E3-Efficient Content Discovery, E4-Efficient Stabilization, R1-Robustness, R2-Censorship-Resistance, P1-Anonymity, P2-Membership-Concealment; score ✓/ X/ ? indicates that a requirement is satisfied/not satisfied/not sufficiently considered, X(?) indicates that requirement is likely not to be satisfied but there is no conclusive proof; Approaches we consider in more detail throughout the thesis are colored cyan

Unstructured approaches inherently fail to fulfill our requirements due to their high communication complexity. Their main application lies in file-sharing with highly popular files stored by many nodes. In contrast, messaging requires the possibility to discover exactly one node at a low complexity. Because unstructured approaches are unable to provide the desired complexity for communication between two uniquely defined entities, we do not consider them in our further analysis.

Virtual overlays, one of two approaches realizing DHT-like overlays in F2F overlays, can offer a communication complexity of $\mathcal{O}(\log^2 n)$ messages. So, while they do not offer efficient routing using only $\mathcal{O}(\log n)$ messages, they are a promising solution for content discovery. However, it remains unclear if the stabilization complexity necessary to maintain such a low routing complexity over an extended period of time is indeed polylog. In Chapter 6, we hence consider the question if virtual overlays can provide polylog routes and polylog stabilization concurrently.

In the context of network embeddings, the existing results indicate that embeddings into low-dimensional l^p spaces are unable to offer the desired polylog routes. Though their unsuitability is merely a conjecture, we exclude those algorithms from our future analysis due to their observed low efficiency and the vast abortive prior work.

In contrast, greedy embeddings based on spanning trees seem highly promising as they provide very short routes. However, they have not been designed and analyzed in the context of F2F networks. In particular, the current design does not consider anonymization, membership-concealment, robustness, and censorship-resistance. Thus, we analyze tree-based greedy embeddings with regard to these requirements in Chapter 7.

In summary, our review of the related work revealed two promising approaches, virtual overlays and greedy embeddings. We base our evaluation of these two approaches upon realistic data sets and a rigorous scientific methodology, which we present before starting our actual evaluation.

Chapter 4

User Models

In Section 2.3, we noted that there are no appropriate user models for F2F overlays, in particular with regard to churn patterns. We thus perform a measurement study in Freenet in order to obtain realistic user models. In addition, evaluating F2F overlays requires social graphs to model the overlay topology. We select these from suitable existing data sets. In the following, we describe these data sets together with the measurement study. The results of the measurement study are published in [134].

4.1 Churn

Our introduction of existing churn models for P2P overlays in Section 2.3.2 indicated the lack of a suitable churn model for F2F overlays. The frequency of arriving and departing nodes is closely related to the overall communication complexity. Hence, a realistic model is essential for evaluating the suitability of algorithms and selecting optimal parameters.

In this section, we obtain a suitable churn model through a measurement study in Freenet. We choose Freenet due to its popularity as an anonymous P2P overlay with a F2F mode. Furthermore, we are in contact with the developers, which might lead to an implementation of our protocols within the Freenet client. In the following, we first describe the measurement methodology and set-up, followed by our results. Last, we validate the correctness of the measured data and name limitations of the study.

4.1.1 Measurement Methodology

In order to explain our methodology, we first need to give some background on specific functionalities of Freenet's implementation. Afterwards, we shortly describe the idea of our method, before specifying the details of our instrumentation of Freenet nodes.

Our methodology for tracing the online status of nodes in the Opennet part of Freenet leveraged Freenet's *FNPRHProbeRequest* message. *FNPRHProbeRequest* allows a user to request some information about a randomly chosen node in the network for the purpose of monitoring and attack detection. A reply to such a request contains one specified information about a random node in the network, e.g., the uptime or distance to neighbors. In particular, *FNPRHProbeRequest* can be parametrized such that the unique *location* of a node u is returned. The responding random node is the endpoint of a random walk with Metropolis-Hastings correction of length 25, so that every node should be selected close to uniformly at random [8]. As nodes can be uniquely identified by their location, the location offers a possibility to trace individual users.

We leveraged the *FNPRHProbeRequest* to track the locations of *all* online users for an extended period of time. Because the responding node is selected randomly, it was impossible to only track a small set of users, as is commonly done in churn measurements, e.g., in [75]. In order to track all users, we repeated queried for unique node *locations*. We inserted monitoring nodes with diverse locations into the overlay. Then, we sent a large number of *FNPRHProbeRequest*s and gathered all replies together with a timestamp. A node was declared offline if we did not receive a reply with its location for a sufficiently long interval. One of the particular challenges of the measurement study was the choice of the interval length.

In order to determine the length of interval without reply after which we declared a node offline, we considered the following trade-off. On the one hand, if the interval is short, the number of false negatives, i.e., declaring an online node offline, is high. By only allowing nodes to remain unresponsive for a short interval, the probability that an online node does not reply increases. Then, one session is divided into multiple short sessions, so that the measured churn rate is higher than the real rate. On the other hand,

if the interval is long, a node might leave the system and come back without being declared offline. By missing short offline periods, the user group might appear more stable than it actually is.

Based on the above considerations, we now formally define our interval choice and our approximation of the session length. In order to decide when a node is declared to be offline, we parametrize our uncertainty in terms of $p \in [0, 1]$, the probability that an online node replies during an interval of length $\tau(p)$. In other words, $1 - p$ is a lower bound on the probability that we declare a node offline despite it being online. So, a node was considered offline if no reply from it had been received for at least time $\tau(p)$. In order to determine $\tau(p)$, let req be a lower bound on the number of received replies per time unit and n be an upper bound on the number of online nodes. Assuming that answering nodes were indeed chosen uniformly at random, the probability that a node did not respond to any of $req \cdot \tau(p)$ requests was

$$1 - p \leq (1 - 1/n)^{req \cdot \tau(p)}. \tag{4.1}$$

Now, we obtained traces by declaring a node online as long as we received a reply at least every $\tau(p)$ time units, otherwise the node was considered to be offline starting from the point of its last reply.

In this manner, we obtained an ordered set $R(u) = \{r_1(u), \ldots, r_{|R(u)|}(u)\}$ with $r_i(u) \in [0, T]$ of reply dates for each user/location u. The start of a session was assumed to be the first time a node had replied after not replying for at least time $\tau(p)$, i.e.,

$$S(u) = \{r_i(u) \in R(u) : i = 1 \text{ or } r_i(u) - r_{i-1}(u) \geq \tau(p)\}.$$

Analogously, the end of a session was defined as the point in time of the last received reply

$$E(u) = \{r_i(u) \in R(u) : i = |R(u)| \text{ or } r_{i+1}(u) - r_i(u) \geq \tau(p)\}.$$

We could estimate the online times of nodes with p balancing our risk of incorrectly declaring nodes offline or online.

4.1.2 Measurement Set-up

The concrete set-up for our study was the following: The measurement was conducted in November 2013 over a period of 9 days using 150 instrumented clients. We obtained traces for roughly 60,000 nodes, of which on average 7,989 were concurrently online. We varied p, the lower bound on the probability that an online node replies within a time $\tau(p)$, between 0.9, 0.925, 0.95, 0.975, 0.99, and 0.999. Our monitoring nodes received at least $req = 10,000$ replies per minute. Choosing $\tau(p)$ according to Equation 4.1 with an estimate of $n = 15,000$ resulted in intervals of roughly 3 ($p = 0.9$) to 10 ($p = 0.999$) minutes as can be seen in Table 4.1. Note that p is a lower bound on the probability to discover a node since we consider a lower bound on req and an upper bound on n.

4.1.3 Churn Data Set

In the following, we subsequently describe the results for the session length, intersession length, and connectivity factor, as defined in Section 2.3.2. Note that we consider the session length, i.e., the time between a node's join and its corresponding departure, to be of particular interest. The time that nodes remain in the system indicates the amount of suitable stabilization complexity. If the sessions are short, stabilization should be fast and inexpensive whereas long sessions indicate that more expensive stabilization algorithms pay off. Figure 4.1 depicts the distributions for various values of p as well as the fitting of the session length for $p = 0.99$.

Session Length: The median session length varied between 49 to 110 minutes, depending on p. In particular, the median session lengths for $p = 0.975$ and $p = 0.99$ were 95 and 99 minutes, respectively. The distribution of the session length is shown in Figure 4.1a.

We fitted the session length distribution to the most common user models, as introduced in Section 2.3.2. In order to determine the most suitable model, we measured the quality of the fit by the residual error of the non-linear least square fit using R [16]. For $p = 0.99$, the fitted distributions, displayed in Figure 4.1b, are the following:

- an exponential distribution with $\lambda \approx 244.738$,
- a Pareto distribution with $\lambda \approx 116.3$ and $k \approx 1.054$,
- a Weibull distribution with $\lambda \approx 186.741$ and $k \approx 0.4788$, and
- a lognormal distribution with $\lambda \approx 97.257$ and $k \approx 0.548$.

p	$\tau(p)$	$\theta(q_i(p))$		
		mean	min	max
0.900	3:27	0.993	0.989	0.996
0.925	3:53	0.993	0.989	0.996
0.950	4:29	0.992	0.989	0.995
0.975	5:31	0.991	0.987	0.994
0.990	6:54	0.989	0.983	0.993
0.999	10:22	0.984	0.979	0.989

Table 4.1: FNProbeRequest Statistics: Time $\tau(p)$ without reply until a node is declared offline, estimation $q_i(p)$ of detecting an online node; cyan row ($p = 0.99$) is the setting used for our churn model

The residual errors were minimized for the Weibull distribution (about $8 \cdot 10^{-3}$). However, the lognormal distribution also resulted in a residual error of only 0.019. The error of the lognormal distribution is mostly due to its underestimation of the fraction of short sessions, as can be seen from Figure 4.1b. Since the session length was underestimated by our measurement methodology in general, the error is acceptable and can be seen as a correction. The fitted Weibull distribution, on the other hand, overestimated the fraction of short sessions, while the exponential and Pareto distribution did not model the shape of the distribution accurately.

Inter-Session Length: The distribution of the inter-session length, i.e., the time between a node departure and the subsequent join of the same node, is displayed in Figure 4.1c. The median inter-session length varied greatly between less than 10 minutes ($p = 0.9$) and close to 6 hours ($p = 0.999$). The reason for this difference was the fact that for 0.9 a lot of sessions were divided into multiple sessions with short inter-sessions. As constant join and leave actions of individual nodes were unlikely [1], we assume that the low median inter-session length for low values of p were an artifact of the measuring methodology. Thus, we consider the results for higher p such as $p = 0.99$ to be more representative, indicating that users leave the network for several hours in general, e.g., over night.

Connectivity Factor: The distribution of the connectivity factor, the overall fraction of the measurement period that nodes remain online displayed in Figure 4.1d, shows that most users were online during a small fraction of the measurement, but also more than 5% of the users had a connectivity factor of nearly 1. Note that in contrast to the session length, the results for the connectivity factor are very close for all p, due to the fact that the overall online time is not largely influenced by splitting one session into multiple sessions. The average connectivity factor is around 0.22, which means that nodes are on average online for more than a fifth of the time.

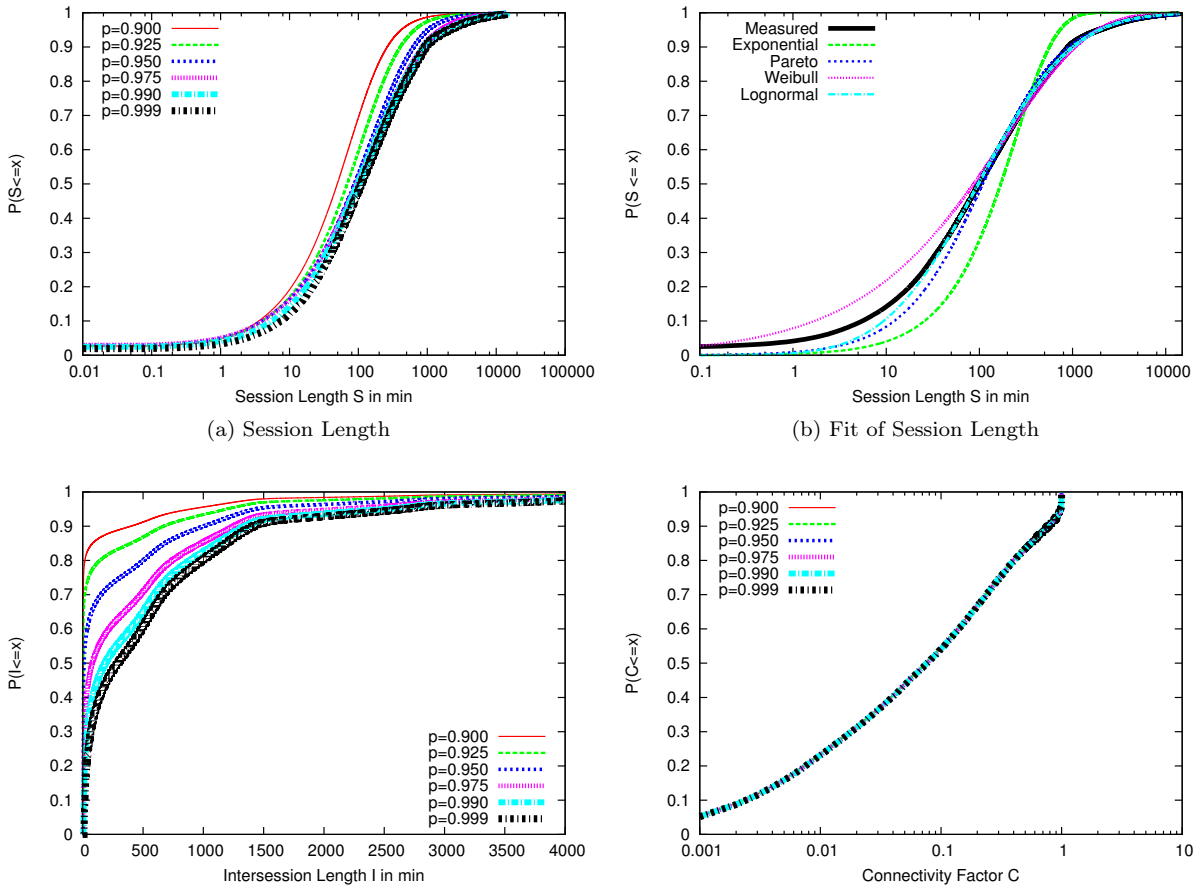
Summary: We discovered that the session length is reasonably well modeled by lognormal or Weibull distributions, but not by a Pareto or exponential distribution. Our median online time of more than an hour for $p \geq 0.95$ is noticeable higher than the online time observed in P2P file-sharing overlays, which varied from 1 to 60 minutes [127]. The result is encouraging because long sessions reduce the stabilization costs and the likelihood of failures due to topology changes. As a consequence, we are positive that structured systems, as evaluated in the following chapters, are sensible solutions for F2F networks.

4.1.4 Validation and Measurement Overhead

We validated that the responding nodes were indeed selected uniformly at random and that the study captured the majority of online nodes. Furthermore, we measured the traffic produced by our measurements in relation to the normal traffic without measurements.

Random Responder Selection: In theory, the (close to) uniform selection of the responding node follows from the application of the Metropolis-Hastings algorithm and the assumption that the Freenet topology is fast mixing [8]. In order to validate this assumption, we obtained a practical validation by considering the distribution of the number of replies sent by our own monitoring nodes. If indeed the choice of the responding node is uniform, the number of replies should be normally distributed. Indeed, the Kolmogorov-Smirnoff test indicates a normal distribution and thus a uniform or close to uniform selection of responding nodes.

[1] As there is no mobile client for Freenet and the service improves if nodes are connected for an extended period of time, such behavior should not occur frequently



(a) Session Length

(b) Fit of Session Length

(c) Inter-session Length

(d) Connectivity Factor

Figure 4.1: Churn data set: Session length for a) all considered p, and b) $p = 0.99$ fitted to common session length models, c) inter-session length, and d) connectivity factor

Node Coverage: We validate that the fraction q of nodes responding during an interval of length $\tau(p)$ is close to 1, i.e., nearly all online nodes in the overlay indeed respond to at least one probe during an interval of time $\tau(p)$. We utilized that the size of a static network can be estimated by sampling two sets of nodes and considering the size of their intersection [99]. In the dynamic overlay Freenet, we obtained a lower bound since the samples were taken at different points in time. Nevertheless, a large intersection between two samples indicated that the majority of the online nodes was included in the sample.

In order to show that we indeed discovered a high fraction of nodes, we split the measurement period into intervals of length $\tau(p)$. We then applied the methodology presented in [99] using the sets of nodes discovered in the respective intervals. In the following, we shortly describe the methodology but refer to [99] for details.

We aimed to estimate the sampling probability q_i, i.e., the probability that an online node responds within the i-th interval. First, we defined a sample A_i to consist of all nodes responding to a probe in interval i. Second, we computed the fraction of the intersection $f_i = \frac{|A_i \cap A_{i+1}|}{|A_i \cup A_{i+1}|}$. Note that the probability that an online node is sampled in interval i and $i+1$ is $q_i q_{i+1}$, and the probability that it is sampled in at least one interval is $1-(1-q_i)(1-q_{i+1})$. For a static overlay and constant q_i, the expected value of f_i is $\mathbb{E}(f_i) = \frac{q_i^2}{1-(1-q_i)^2}$. We hence obtained an estimate $\theta(q_i) = \frac{2f_i}{1+f_i}$ of q_i from $f_i = \frac{q_i^2}{1-(1-q_i)^2}$. In this manner, we estimated the sampling probability q_i and hence the expected fraction of nodes we sampled.

The results of the above analysis shows that we indeed saw most online nodes within an interval of length $\tau(p)$. Consider the values for mean, minimal, and maximum $\theta(q_i)$ displayed in Table 4.1. With exception of the minimum for $p = 0.999$, all estimates of the sampling probability exceeded 0.98 despite the existence of network dynamics. So, we indeed detected the majority of online nodes within an interval of the length $\tau(p)$ for large enough p.

Measurement Traffic: We measured the average number of probes from our nodes that were forwarded in comparison to the average number of content requests forwarded. The average number of content requests a node has to process in one hour under normal conditions was 13,000. Each request requires

both relaying the request and its answer, adding up to a total of 27,000 messages. In addition, there is a considerable amount of traffic for stabilization, such as heartbeat messages and node announcements. In contrast, the average overhead produced by our study is about 2,000 messages per node and hour, which makes up a noticeable but not large fraction of the total traffic.

The above argumentation indicates that our measured data is indeed largely accurate and the load produced by our measurement study was not exhaustive enough to disrupt normal communication.

4.1.5 Limitations

Our study has the following limitations:

- The user models are based upon the current user group of Freenet. There are no guarantees that future users of F2F overlays exhibit the same characteristics. However, we consider integrating our protocols in Freenet. So, our initial user group should exhibit similar characteristics.
- The user models are based upon all Freenet users, including mostly Opennet users rather than F2F overlay users. While we expect a similar user group is interested in participating in F2F overlays, we cannot guarantee that the observed characteristics are valid in a pure F2F overlay.
- By defining a session as the time between the first and last reply of the node within the session, we clearly underestimate the session length, disregarding the time before the first and after the last encounter. However, underestimating entails a more aggressive churn model, so that algorithms achieving a suitable performance within this model are likely to perform well in less extreme scenarios.

Despite these limitations, we consider the gathered data set to be sufficient for our purpose: complementing theoretical bounds and providing an rough estimate of the degree of necessary stabilization.

4.2 Social Graphs

F2F overlay topologies correspond to social graphs, namely graphs consisting of users and their real-world trust relationships. The communication overhead as well as the resilience of the overlay is thus closely related to the structure of the social graphs. As a consequence, the selection of suitable graphs is essential for the applicability of our results. In the following, we first describe the origin and post-processing of the selected graphs before evaluating them with regard to their degree and shortest paths length distribution, which we introduced in Section 2.1.3.

Throughout this thesis, we utilize three different graphs, corresponding to three different scenarios. First, we consider a subgraph of the multi-purpose online social network (OSN) Facebook (denoted *FB* in the following), namely the New Orleans regional network [5]. In Facebook, a link between nodes indicates a friendship relation within the social network, thus incorporating the concept of F2F network. However, it is known that Facebook friends frequently do not share a close trust relation [35], indicating that the degree of a node is possibly overestimated in comparison to a F2F overlay. In contrast, the Studentenportal Ilmenau is a local special-purpose OSN of the TU Ilmenau in Germany, directed towards students and their interaction within the university. Similar to *FB*, the social graph in Student's Portal Ilmenau (denoted *SPI* in the following) represents friendship relations of users [119]. Contrary to *FB*, the degree in *SPI* is likely to underestimate the connections in a F2F overlay because only friends within the context of the university are included in *SPI*. Our snapshot of *SPI* is from October 2011 and contains the complete OSN topology. In addition to OSNs, we consider trust graphs provided in the context of the Web-of-Trust [24]. Here, a link from a node u to a node v indicates that u signed v's public key by which u supposedly verifies v's identity. However, it is unclear how signing someone's key and real-world trust relationships are related. Often, signing a key merely indicates an acquaintance, especially since the emergence of key-signing parties. Nevertheless, Web-of-Trust topologies represent an interesting case study, because they are associated with realizing secure communication based on social links. In this thesis, we make use of a Web-of-Trust snapshot from January 1st, 2011 (denoted *WOT* in the following). We post-process all snapshots by removing unidirectional links because F2F overlays require a mutual trust relation. Furthermore, we remove nodes not contained in the giant connected components, usually individual nodes or pairs, to ensure that routing between arbitrary nodes is indeed possible.

In the following, we compare the three topologies with regard to their i) size, ii) degree distribution (Equation 2.1), and iii) shortest path length distribution (Equation 2.2). It has been shown that the degree distribution of the social graph is closely related to the resilience of the overlay [51, 52]. Furthermore, the shortest paths present a lower bound on the length of routes discovered by the algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$. For this reason, those two metrics are essential for setting our results into perspective.

Metric	FB	SPI	WOT
Nodes	63392	9222	41688
Mean Degree	25.77	10.58	13.55
Median Degree	11	7	4
Maximal Degree	1098	147	1929
Mean Shortest Path Length	4.32	4.67	5.01
Diameter	15	12	16

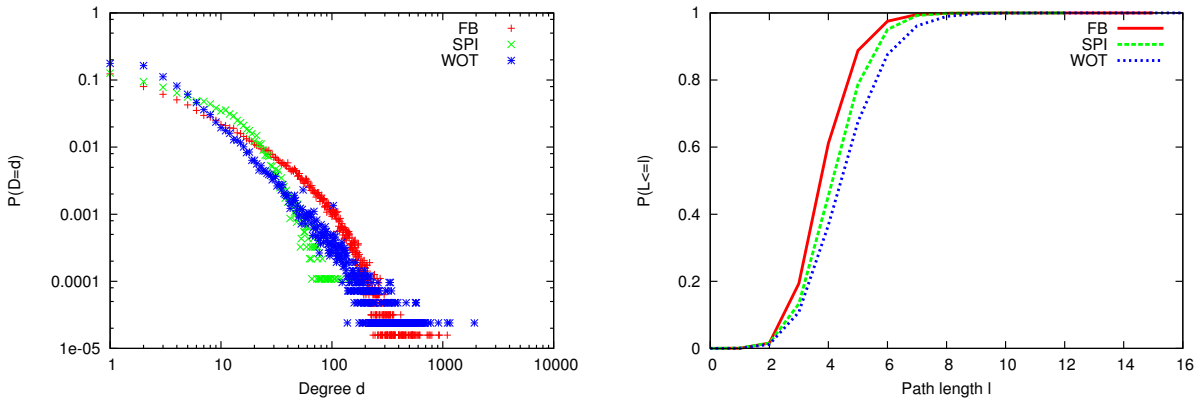
Table 4.2: Scalar Metrics of Social Graphs

Size: The number of nodes and edges in the three graphs vary, as displayed in Table 4.2. *SPI* has a comparable small user group, corresponding to a graph with less than 10,000 nodes. The number of users is typical for special-purpose social networks or social networks in their early stages. Indeed, *SPI* exhibits multiple properties commonly associated with social networks in early developing stages [119]. Thus, *SPI* is particularly useful for estimating the performance of our algorithms soon after publishing the first implementation. Achieving a good performance is essential during this early development stage, as users are likely to cease using the software otherwise. In contrast, *WOT* and *FB* present more stable networks of a larger size, namely with more than 40,000 and 60,000 users, respectively. In particular, the latter corresponds to the current size of the Freenet system [134]. Hence, these graphs are of a realistic size for current F2F overlays.

Degree Distribution: The degree distributions, displayed in Figure 4.2a, emphasize the differences among the three topologies. As can be seen in Table 4.2, *FB* exhibits a much higher mean degree than the other two graphs. In particular, *SPI*, restricted to a small fraction of a user's overall contacts, only achieves an average degree of around 10.

In contrast to the OSNs, *WOT*'s degree distribution is more skewed than the others, as can seen from the high fraction of nodes with degree one and the comparable high number of high degree nodes in Figure 4.2a, Indeed, *WOT* contains roughly 40,000 nodes, which is more than four times the number of nodes in *SPI*. Yet, the medium degree in *WOT* is only 4, in contrast to 7 in *SPI* and 11 in *FB*. However, *WOT*'s maximal degree is close to 2,000, nearly twice as high as the maximal degree in *FB* despite the latter's larger size of more than 60,000 nodes.

Note that nodes with a low degree are unlikely to find alternative routes if one of neighbors does not response adequately. So, we expect that *WOT* and *SPI* exhibit a lower robustness and possibly censorship-resistance. However, the stabilization overhead should be highest for *FB*, because *FB* exhibits the highest number of nodes and links. Thus, it can expected that the number of nodes affected by a topology change is high as well.



(a) Degree (b) Shortest Path (CDF)

Figure 4.2: Degree and shortest path length distributions of graph data sets

Shortest Path Length Distribution: The differences in the degree distributions also impact the length of the shortest paths, displayed in Figure 4.2b. If the average number of neighbors is higher, an increased number of nodes can be reached in few steps.

Hence, though being the largest graph, *FB* exhibits the shortest paths, as can be seen both from the average shortest path length in Table 4.2 as well as from the cumulative distribution function in Figure 4.2b. While the diameter of *SPI* is lower than for the other graphs due to its small size, Figure 4.2b indicates that the number of longer paths in *FB* is negligible and thus unlikely to have a considerable impact on the average number of messages required for node or content discovery.

Despite the existence shorter path in *FB*, we do not necessarily expect a lower overhead for routing in *FB* than in *SPI* and *WOT*. Though *FB* exhibits the shortest paths, the high average degree indicates that there is a high probability of choosing a non-optimal neighbor during routing. Such an increased probability to 'take a wrong turn' indicates that length of the discovered routes and the shortest path length in *FB* are likely to differ more drastically than in the other two graphs. As a consequence, it might be that *SPI* and *WOT* exhibit a lower routing overhead despite longer shortest paths. So, it is highly dependent on the quality of the routing algorithm if it discovers the shortest routes in *FB* or one of the other graphs.

Overall, shortest paths of 4 to 5 hops are very encouraging. If indeed our algorithms can detect routes of a similar length, even real-time communication is possible, as it is for example when using 3-hop anonymization in Tor.

4.3 Discussion

Because none of our selected data sets have been gathered in a pure F2F overlay, we cannot prove that they represent F2F overlays sufficiently well. However, our social graphs cover a wide range of scenarios. The graph of a real-world F2F overlay is likely to be a trade-off of these scenarios. Thus, a good performance on all exemplary scenarios of them is a valid indicator that our algorithms is indeed suitable for F2F overlays. Our churn patterns are based upon a large-scale measurement study and approximate the actual behavior of users during the measurement period very closely. As we our protocols could be implemented in the Freenet client, the measurement study represent a close approximation of our initial user group. Thus, we are confident that we selected appropriate realistic data sets for our evaluation.

Chapter 5

Methodology

In this chapter, we introduce our methodology for evaluating F2F overlays. Throughout the thesis, we make use of two evaluation methods, mathematical analysis and simulation. We use mathematical analysis to provide asymptotic bounds on the communication complexity and to assess the scalability of algorithms. In addition, we evaluate our anonymization algorithms theoretically to determine upper bounds on the probability of identifying communicating parties. Complementing the theoretical bounds, the simulation-based evaluation validates the theoretical bounds and provides concrete results. We decided on these evaluation methods in accordance with the goals of the thesis: providing a better characterization of existing F2F approaches as well as techniques enabling such a characterization, and designing a conceptual approach that can principally satisfy the eight requirements defined in Section 2.6.

Our main contribution in this chapter concerns the evaluation of efficiency and scalability, primarily their mathematical modeling and analysis. Here, we extend the existing methodology to deal with embeddings that are not greedy. Furthermore, we model routing and stabilization complexity as time-dependent correlated processes to evaluate how restricting one of them affects the other over a long period of time. Afterwards, we validate our new methodology. In addition to efficiency and scalability, we also present our methodology for robustness and censorship-resistance as well as anonymity and membership-concealment, albeit in a shorter manner.

For each of the three evaluation aspects, we first give a classification of algorithms. Afterwards, we motivate our focus on specific classes of algorithms. For these classes, we formally define evaluation metrics. Last, we describe how to derive bounds on these metrics, starting with the state-of-the-art method and adding modifications if applicable.

In Section 5.1, we present common notions in the area of probability theory. Based on these notions, we explain our methodology regarding efficiency and scalability in Section 5.2 and its validation in Section 5.3. Afterwards, we consider robustness and censorship-resistance in Section 5.4. In Section 5.5, we present how we evaluate anonymity and membership-concealment. The proposed methodology has been applied in our publications as stated in the respective sections, and the results in Section 5.3 are partially published in NetSys 2015 [131].

5.1 Probability Theory

Throughout this thesis, we model routing and stabilization complexity in terms of random variables and stochastic processes. The reason for this probabilistic model is to abstract from a concrete social graph and its evolution over time to a more general characterization. Thus, we deepen our discussion of probability theory initialized in Section 2.1. We start by defining important general notions and results. Afterwards, we define the concept of a system model in terms of the presented notions.

5.1.1 Random Variables and Stochastic Processes

Our evaluation relies upon multiple concepts from the area of probability theory, which we summarize in the following. We merely state the results, a general introduction to probability theory, including the proofs of the presented results, can be found in e.g., [32].

Probability Space: Probability theory builds upon the notation of a probability space. All remaining results presented in this section follow from the properties of probability spaces.

We first need to give the definition of a σ-Algebra. Let $\mathcal{P}(A)$ denotes the power set of the set A, i.e., the set of all subsets of A. A σ-Algebra $\mathcal{A}$ of A is a subset of $\mathcal{P}(A)$, so that i) A and the empty set $\emptyset$

are contained in $\mathcal{A}$, ii) complements of sets in $\mathcal{A}$ are also in $\mathcal{A}$, and iii) infinite unions of sets in $\mathcal{A}$ are in $\mathcal{A}$. The definition of a probability space is a σ-Algebra equipped with a function P satisfying three properties.

Definition 5.1. *A* probability space $(\Omega, \mathcal{A}, P)$ *consists of a nonempty set* Ω*, a* σ*-Algebra* $\mathcal{A}$ *of* Ω *and a function* $P : \mathcal{A} \to [0, 1]$*, so that:*

1. $P(\Omega) = 1$
2. *For* $A, B \in \mathcal{A}$ *with* $B \subset A$*:* $P(A \setminus B) = P(A) - P(B)$
3. *For pairwise disjunct sets* $A_1, A_2, ... \in \mathcal{A}$*:* $P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$

An element of $\mathcal{A}$ *is called an* event *and* P *is a* probability mass *or* probability mass function.

Random Variables: A generalization of a probability space is a measure space $(\Omega, \mathcal{A}, \mu)$ such that μ is only required to fulfill the last two conditions in Definition 5.1. Then, an element of $\mathcal{A}$ is referred to as a *measurable set*. A *measurable* function $f\colon (\Omega_1, \mathcal{A}_1, \mu_1) \to (\Omega_2, \mathcal{A}_2, \mu_2)$ is a function between two measure spaces such that the preimage of any measurable set is also a measurable set, i.e.,

$$\forall A_2 \in \mathcal{A}_2 : f^{-1}(A_2) = \{a_1 \in A_1 : f(a_1) \in A_2\} \in \mathcal{A}_1.$$

The notion of measurable functions allows us to define random variables and stochastic processes, the decisive concepts underlying our theoretical analysis. Intuitively, random variables describe the possible outcomes of one experiment. Formally, they are defined as measurable functions on a probability space.

Definition 5.2. *A* random variable X *on a probability space* $(\Omega, \mathcal{A}, P)$ *is a* $\mathcal{A}$*-measurable function* $X : \Omega \to \Omega_2$ *for a measure space* $(\Omega_2, \mathcal{A}_2, \mu_2)$*. The random variable is real-valued if* $\Omega_2 = \mathbb{R}$ *with the Borel* σ*-Algebra, i.e., the smallest algebra including all open sets, and its standard measure, which assigns each interval* $(a, b) = \{r : a < r < b\}$ *its length* $b - a$*.*

For a random variable X and $\mathcal{A}_2$-measurable set A_2, we can determine the probability of the event $\{\omega \in \Omega : X(\omega) \in A_2\}$. For brevity, we write $P(X \in A_2)$ for $P(\{\omega \in \Omega : X(\omega) \in A_2\})$. Note that a random variable X defines a new probability mass function P_X on $(\Omega_2, \mathcal{A}_2)$ with $P_X(A_2) = P(X \in A_2)$.

In addition to dropping the braces indicating events, we use the following conventions:

- The probability of the intersection of two events A and B is denoted by $P(A, B)$ rather than $P(A \cap B)$.
- The *conditional probability* of an event A given the event B with $P(B) > 0$ is

$$P(A|B) = \frac{P(A, B)}{P(B)}.$$

- Let X, Y be random variables, r be a real number, and *op* be a relation, most commonly $=, <, >$. For brevity, we write $P(X \; op \; r)$ instead of $P(\{\omega \in \Omega : X(w) \; op \; r\})$.
- For a random variable X, $F_X(x) = P(X \leq x)$ denotes the *cumulative distribution function.*

Now, we shortly give the definitions of independence, identical distributions, and expected value. Note that we focus on discrete random variables, i.e., we assume $P(X \in \mathbb{Q}) = 1$ for the set of rational numbers $\mathbb{Q}$.

Definition 5.3. *Two real-valued random variables* X, Y *are* independent *if for all sets* I, J $P(X \in I, Y \in J) = P(X \in I)P(Y \in J)$*.* X *and* Y *are called* identically distributed *if* $P(X \in I) = P(Y \in I)$ *for all* I*.*

The expectation of a discrete random variable X *is given by* $\mathbb{E}(X) = \sum_{x \in \mathbb{Q}} xP(X = x)$*. Furthermore, the* conditional expectation *of* X *given an event* B *with* $P(B) > 0$ *is* $\mathbb{E}(X|B) = \sum_{x \in \mathbb{Q}} xP(X = x|B)$.

The generic definition for the expected value of random variables relies on additional background on measure theory. However, the simplified definition suffices for our purposes.

Stochastic Processes: While random variables describe the possible outcome of one experiment, stochastic processes describe the outcome of a sequence of experiments.

Definition 5.4. *A sequence $(X_i)_{i\in\mathbb{N}_0}$ of random variables is called a (discrete)* random *or* stochastic process.

Whereas Definition 5.4 does not assume any dependence between the random variables X_i of a stochastic process, such a dependence usually exists. As a consequence, we express the probability distribution of X_i in terms of the random variables $X_0, \dots, X_{i-1}$. For simplicity, assume that all random variables share a common image S. We then derive the probability distribution of X_i by *conditioning* on $X_0, \dots, X_{i+1}$, i.e.,

$$P(X_i = x_i) \tag{5.1}$$
$$= \sum_{x_0,\dots,x_{i-1}\in S} P(X_i = x_i | X_0 = x_0, \dots, X_{i-1} = x_{i-1}) P(X_0 = x_0, \dots, X_{i-1} = x_{i-1}).$$

In this context, *Markov chains* are of particular importance, since they offer a straight-forward analysis. So, we aim to approximate processes as Markov chains if possible. A stochastic process $(X_i)_{i\in\mathbb{N}_0}$ is a Markov chain if the variable X_i given $X_0, \dots, X_{i-1}$ only depends on its immediate precedent state X_{i-1}, i.e., in Equation 5.1

$$P(X_i = x_i | X_0 = x_0, \dots, X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1}). \tag{5.2}$$

In addition to Markov chains, we consider *increasing* or *decreasing* stochastic processes, i.e., $X_i \geq X_{i-1}$ or $X_i \leq X_{i-1}$ for all $i \geq 1$.

Additional Results: Immediate consequences from the definition of conditional probability frequently applied in probability theory are Bayes' rule and the rule of total probability.

Lemma 5.5. *For two events A, B with $P(A) > 0, P(B) > 0$, we have*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \text{ (Bayes' Rule).}$$

Furthermore, if $A = \bigcup_{i=1}^{m} A_i$ is the union of pairwise disjunct *events A_i, i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$, we have that for each A_i*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{m} P(B|A_j)P(A_j)} \text{ (Total Probability).}$$

In addition, we have the following relation for the conditional probability of events A, B, C

$$P(A|B,C) = \frac{P(A,B|C)P(C)}{P(B,C)}.$$

All of the above results are implicitly used throughout the thesis.

5.1.2 System Models

In this thesis, we are usually not concerned with one specific graph or system but with a class of systems with certain characteristics. For example, rather than determining the communication complexity in one specific instance of a F2F overlay, we want to derive the expected complexity for all systems following a specific design.

We define a system by a pair $\mathbf{Sys} = (G, P)$ of a graph $G = (V, E)$ and a set P of *node properties* or *weights* $pro : V \to \mathbf{X}_{pro}$. These properties can be interpreted as local state information at each node. Throughout this thesis, we only require node properties, but edge properties are a straight-forward extension. In order to model the computation of certain *characteristics* or *metrics* of graph models, let $\mathbb{SYS}_n^{\mathbf{X}_1,\dots,\mathbf{X}_r}$ be the set of all graphs with n nodes and properties $pro_i : V \to \mathbf{X}_i$. A *system model* is a pair of random variables $\mathbf{SM} = (\mathbf{GM}, \mathbf{PM})$ with values in $\mathbb{SYS}_n^{\mathbf{X}_1,\dots,\mathbf{X}_r}$, illustrated for $n = 2$ and one binary property in Figure 5.1. So, the random vector $\mathbf{SM}$ defines a common probability distribution of both the topology of a system and its properties.

We now define a (deterministic) *metric* or *characteristic* of a system model $\mathbf{SM}$ as a function

$$W : \mathbb{SYS}_n^{\mathbf{X}_1,\dots,\mathbf{X}_r} \to \mathbb{R}.$$

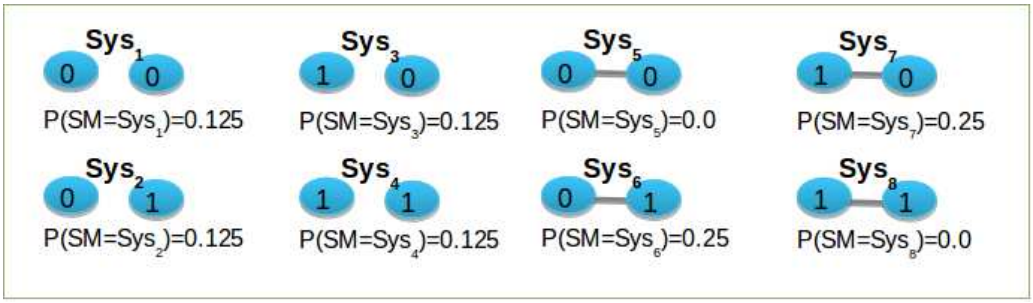


Figure 5.1: Examples of a system model on the set $\mathbb{SYS}_2^{\{0,1\}}$, i.e., the set of all graphs with two nodes and a binary node property assigning each node either the value 0 (e.g., honest) and 1 (e.g., attacker). The system model **SM** assigns each of the eight systems a probability.

So, the expected value of the metric is

$$\mathbb{E}(W) = \sum_{\mathbf{Sys} \in \mathbb{SYS}_n^{\mathbf{X}_1,\ldots,\mathbf{X}_r}} W(\mathbf{Sys}) P(\mathbf{SM} = \mathbf{Sys}). \tag{5.3}$$

If W is a non-deterministic function and $W(\mathbf{Sys})$ a random variable, we replace the value $W(\mathbf{Sys})$ with its expectation $\mathbb{E}(W(\mathbf{Sys}))$. For multi-scalar metrics $\mathbf{W} = (W(1), \ldots, W(l))$, we define the expectation of $\mathbf{W}$ as the vector $\mathbb{E}(W) = (\mathbb{E}(W_1), \ldots, \mathbb{E}(W_l))$ with $\mathbb{E}(W_i)$ defined as in Equation 5.3.

We plan to derive asymptotic bounds on routing and stabilization complexity by defining F2F overlays as system models. For the purpose, we assume that the graph model **GM** only selects connected graphs with a logarithmic diameter. The properties of the system and thus the random variable **PM** depend on the specific approach, i.e., the local state maintained by the approach. We then define the communication complexity as a metric W and derive the expected value as described in Equation 5.3. In the following sections, we define our set of metrics and describe how to derive the expectation of each of these metrics.

5.2 Efficiency and Scalability

We start by introducing the methodology for deriving bounds on routing and stabilization complexity, detailing both the state-of-the-art methodology and our adaption thereof. Afterwards, we combine both methods in order to evaluate the total communication complexity in a dynamic overlay. Last, we introduce our simulation model for complementing the asymptotic bounds with the concrete *overhead*, i.e., the actual number of messages required in certain scenarios.

5.2.1 Communication Complexity of Routing Algorithms

In this section, we consider how to evaluate the communication complexity of the routing algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$ for node and content discovery, respectively. The communication complexity is defined as the expected value of a metric with regard to a system model $\mathbf{SM} = (\mathbf{GM}, \mathbf{PM})$, following our definitions from Section 5.1.2. As the methodology is similar for both algorithms, we consider a generic algorithm $\mathbf{R}$ for routing a request from a *source node* s to a *destination* or *target node* e. Note that the request can be either explicitly addressed to e or concern content stored at e.

Algorithm Classification: There are various categories for classifying routing algorithms. Here, we classify algorithms according to the following criteria:

- Single-threaded or parallel: We call $\mathbf{R}$ single-threaded if nodes forward each request to at most one neighbor. Otherwise, we call $\mathbf{R}$ parallel.
- Greedy or non-greedy routing: In general, an algorithm is called greedy if it solves a problem stepwise selecting the most promising option in each step. In this thesis, we narrow this definition to only refer to the standard greedy routing algorithm, which forwards a request to the neighbor closest to the target according to some distance and terminates if a node has no closer neighbor. We refer to all other algorithms as non-greedy or variants of greedy routing. The reason for this restriction of the common definition lies in the existence of vast related work on this standard algorithm, whereas alternative algorithms are largely disregarded, at least with regard to theoretical analysis, as discussed below.

- Deterministic or in-deterministic: The algorithm $\mathbf{R}$ is deterministic if its behavior is fully determined by the system **Sys**, the source node s, and the target information. Otherwise, $\mathbf{R}$ is in-deterministic.

Clearly, the evaluation of the algorithm depends on its classification with regard to the above algorithm. Existing approaches for routing in F2F overlays can belong to any of the above classes, as can be seen from our survey in Section 3. Unstructured overlays use (in-deterministic) parallel routing, whereas Freenet uses a deterministic single-threaded algorithm. Virtual overlays rely on deterministic routing algorithms but can be either single-threaded (X-Vine) or parallel (MCON). The fact that only greedy embeddings, which have not yet been evaluated in the context of F2F overlays, rely on the standard greedy algorithm indicates the need for evaluation techniques that can deal with non-greedy embeddings.

Evaluation Metrics: We start by formally defining the communication complexity of $\mathbf{R}$, first in terms of a source-destination pair (s, e) and then as an average over all such pairs. Let $CR^{\mathbf{R}}_{s,e}$ be the number of messages $\mathbf{R}$ requires to route a request from s to e. In terms of system models, $CR^{\mathbf{R}}_{s,e}$ is a metric and we compute its expected value as follows. If $\mathbf{R}$ is a deterministic algorithm, $CR^{\mathbf{R}}_{s,e}$ is a constant $cr_{\mathbf{Sys}}$ for each system $\mathbf{Sys} \in \mathbb{SYS}_n^{\mathbf{X}_1,\dots\mathbf{X}_r}$. As a result, we have

$$\mathbb{E}\left(CR^{\mathbf{R}}_{s,e}\right) = \sum_{\mathbf{Sys}\in\mathbb{SYS}_n^{\mathbf{X}_1,\dots\mathbf{X}_r}} cr_{\mathbf{Sys}} P(\mathbf{GM} = \mathbf{Sys}). \tag{5.4}$$

Otherwise, if $\mathbf{R}$ is in-deterministic, $cr_{\mathbf{Sys}}$ is a random variable rather than a constant. Then, we compute the expected value of $CR^{\mathbf{R}}_{s,e}$ as in Equation 5.4 but replace $cr_{\mathbf{Sys}}$ with its expected value $\mathbb{E}(cr_{\mathbf{Sys}})$. Now, we define the *expected communication complexity* of $\mathbf{R}$ by averaging over all distinct source-destination pairs (s, e), i.e.,

$$\mathbb{E}(CR^{\mathbf{R}}) = \frac{1}{n(n-1))} \sum_{s,e\in V: s\neq e} \mathbb{E}(CR^{\mathbf{R}}_{s,e}). \tag{5.5}$$

In addition to the communication complexity, the length of the shortest discovered route $R^{\mathbf{R}}$ between source and destination is of interest. $R^{\mathbf{R}}$ is closely related to the delay experienced by the communication partners. Analogously to Equation 5.5, we define the *expected routing length* or *expected hop count* as

$$\mathbb{E}(R^{\mathbf{R}}) = \frac{1}{n(n-1))} \sum_{s,e\in V: s\neq e} \mathbb{E}(R^{\mathbf{R}}_{s,e}). \tag{5.6}$$

with $R^{\mathbf{R}}_{s,e}$ denoting the length of the shortest discovered route from s to e restricted to requests that were successfully delivered. Note that for a single-threaded algorithm the communication complexity and the routing length are equal.

State-of-the-Art Methodology: Most of the state-of-the-art research on the complexity of decentralized routing algorithms considers a single-threaded greedy algorithm and assumes a greedy embedding. In the following, we describe their methodology. The details of the derivation then depend on the system model **SM**. Prominent examples for applications of the presented methodology are Kleinberg's model [86] and its extensions [66, 67, 91, 100, 101].

The underlying assumption of the approach is that nodes are assigned coordinates from a metric space, i.e., one of the system properties is an embedding $id : V \to M$ for a metric space M. We model routing as a Markov chain $(X_i)_{i\in\mathbb{N}_0}$ such that X_i gives the distance of the i-th node on the routing path to the destination e. Then the routing length corresponds to the number of steps until the process $(X_i)_{i\in\mathbb{N}_0}$ reaches 0. Based on the system model **SM** and the routing algorithm $\mathbf{R}$, we obtain a lower or upper bound on probability

$$P(X_{i+1} = d' | X_i = d, X_{i-1}, \dots, X_0). \tag{5.7}$$

After characterizing the process in this manner, results from probability theory are applied to obtain the desired bounds. Figure 5.3a illustrates a realization of the stochastic process $(X_i)_{i\in\mathbb{N}_0}$ for a routing length of 3.

The applied results from probability theory usually rely on two properties of the process $(X_i)_{i\in\mathbb{N}_0}$:

1. Greedy routing can be applied, hence the process $(X_i)_{i\in\mathbb{N}_0}$ is decreasing.

2. The process is approximately a Markov chain.

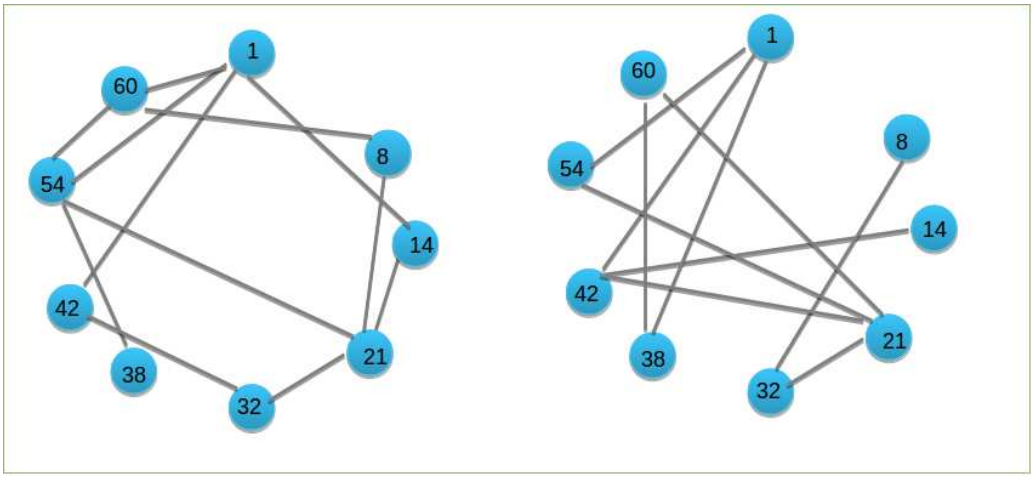


Figure 5.2: Two examples for non-greedy embeddings in a ring of length 64. left: The embedding exhibits only local inaccuracies, corresponding connections to at least every second closest node rather than to the closest node. right: The embedding does not guarantee that all nodes have close neighbors, for example node 8 and 14 only have neighbors that are far in terms of their coordinates' distance.

The first assumption is particularly problematic in the context of F2F overlays. F2F overlays might not provide a greedy embedding either due to the embedding algorithm itself or due to node failures temporarily destroying the structure of the embedding. So, an alternative routing algorithm might be used. For instance, the Freenet algorithm does not correspond to a decreasing process, as detailed in Section 3.2. Thus, we now show how to modify the above methodology to deal with non-greedy embeddings.

Non-greedy Embeddings: We differentiate two types of non-greedy embeddings:

1. The embedding has some slight local "inaccuracies" such as in the generalized Kleinberg model introduced in Section 3.2. In other words, nodes share links with nodes that are close with regard to their coordinates but may not share links with those nodes closest in distance. Thus, the distance to the destination generally decreases but might increase at the very end of the routing.

2. There are nodes for which all neighbors are distant, so that they cannot forward a request to any closer node despite being far from the destination.

Figure 5.2 illustrates the two scenarios. In the following, we describe how to obtain asymptotic bounds for them.

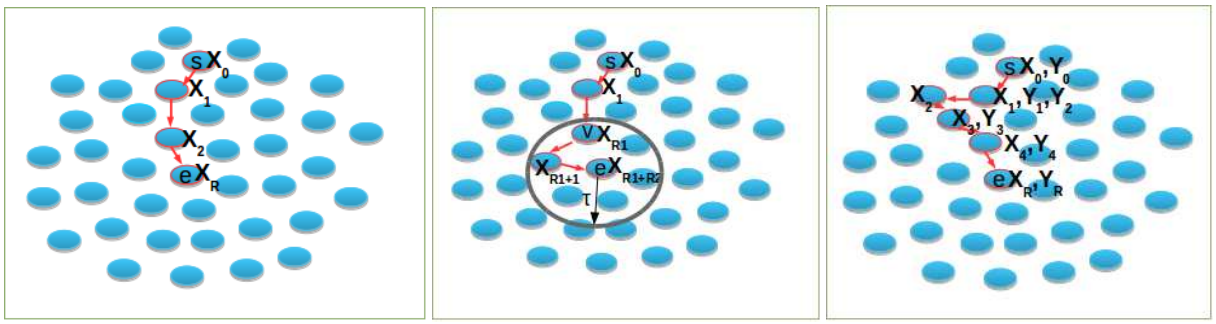
We start by considering the first scenario. We aim to determine the expected value $\mathbb{E}(R^{\mathbf{R}}_{s,e})$ based on the stochastic process $(X_i)_{i\in\mathbb{N}_0}$ describing the distance of the i-th node on the path to the destination. If the process $(X_i)_{i\in\mathbb{N}_0}$ is decreasing but for the last steps of the routing, we divide the routing into two phases: the first exhibits a monotone decrease in distance whereas the second does not. Let τ be the maximal distance such that the $(X_i)_{i\in\mathbb{N}_0}$ is guaranteed to be decreasing. In other words, τ is chosen such that if $X_i \geq \tau$, then $X_i > X_{i+1}$. $R1^{\mathbf{R}}_{s,e}$ denotes the number of messages sent until the requests reaches the first node v within distance τ of the destination e. Similarly, $R2^{\mathbf{R}}_{s,e}$ denotes the number of messages sent after v is reached. Thus, we determine the expected value $\mathbb{E}(R^{\mathbf{R}}_{s,e})$ as the sum of the expected values $\mathbb{E}(R1^{\mathbf{R}}_{s,e})$ and $\mathbb{E}(R2^{\mathbf{R}}_{s,e})$, as illustrated in Figure 5.3b.

During the first phase, normal greedy routing is used. Thus, we apply the standard methodology to determine bounds on the number of steps $\mathbb{E}(R1^{\mathbf{R}}_{s,e})$.

In the second phase, we determine the number of steps $\mathbb{E}(R2^{\mathbf{R}}_{s,e})$ to reach the destination e starting from v, the first node on the routing path within distance τ of the destination. The derivation of $\mathbb{E}(R2^{\mathbf{R}}_{s,e})$ is highly dependent on the routing algorithm. We present one method to determine upper and lower bounds but do not claim its general applicability.

For an upper bound, the following approach can be applied: In the first step, we prove that for a (sub-)graph of m nodes, routing is guaranteed to terminate in $g(m)$ steps. In the second step, we consider the event B that there exists a node u that receives the request in the second phase with $\delta_X(id(u), id(e)) \geq d_{max}$ for some d_{max}. Our goal is to determine an upper bound on $\mathbb{E}(R2^{\mathbf{R}}_{s,e})$ by conditioning on B, i.e.,

$$\mathbb{E}(R2^{\mathbf{R}}_{s,e}) = \mathbb{E}(R2^{\mathbf{R}}_{s,e}|B)P(B) + \mathbb{E}(R2^{\mathbf{R}}_{s,e}|B^{\perp})P(B^{\perp}) \leq \mathbb{E}(R2^{\mathbf{R}}_{s,e}|B)P(B) + \mathbb{E}(R2^{\mathbf{R}}_{s,e}|B^{\perp}). \tag{5.8}$$



(a) Greedy (b) 2 Phases (c) Closest Node

Figure 5.3: Illustration of methods for deriving the routing length R of a single-threaded algorithm relying on node coordinates from a metric space. Routing from source s to destination e corresponds to a stochastic process $(X_i)_{i\in\mathbb{N}_0}$ describing the i-th node's distance to e in terms of the coordinates. We consider three scenarios: a) The distance is a monotonously decreasing stochastic process. b) Routing consists of 2 phases. First, the distance is decreasing until a node v within distance τ of e is reached. When within distance τ, there might be slight increases in distance due to local inaccuracies of the coordinate assignment. Thus, the routing length R derived as sum of routing length $R1$ and $R2$ of the two phases. c) There are no guarantees for a decreasing distance. In order to nevertheless use the results for such process, we define a second decreasing process $(Y_i)_{i\in\mathbb{N}_0}$, which gives the distance of the closest node of the first i nodes on the path. In the above example, the first node remains the closest seen node after the second step, hence X_1, Y_1, and Y_2 are all given by the distance of the same node.

The first step of the proof ensures that the routing terminates in $g(n)$ steps, so $\mathbb{E}(R2^{\mathbf{R}}_{s,e}|B) = \mathcal{O}(g(n))$. In order to prove that the term $\mathbb{E}(R2^{\mathbf{R}}_{s,e}|B)P(B)$ is bound by a constant, we show

$$P(B) = \mathcal{O}\left(\frac{1}{g(n)}\right).$$

Otherwise, if B does not hold, the routing during the second phase is restricted to the $m = br(d_{max})$ nodes within distance d_{max} of e. By the first part of the proof, routing terminates in $g(m)$ steps. Consequentially, it follows from Equation 5.8 that the expected value of R_2 is bound from above by

$$\mathbb{E}(R2^{\mathbf{R}}_{s,e}) = \mathcal{O}\left(\frac{1}{g(n)}g(n)\right) + \mathcal{O}\left(g(m)\right) = \mathcal{O}\left(g(br(d_{max}))\right).$$

We derive a lower bound on $\mathbb{E}(R2^{\mathbf{R}}_{s,e})$ by considering an event S, so that i) the probability $P(S)$ is constant in the network size, and ii) if S holds, the routing length is at least $h(n)$, i.e., $\mathbb{E}(R2^{\mathbf{R}}_{s,e}|S) = \Omega\left(h(n)\right)$. As $P(S)$ is constant, the asymptotic bound

$$\mathbb{E}(R2^{\mathbf{R}}_{s,e}) = \Omega\left(\mathbb{E}(R2^{\mathbf{R}}_{s,e}|S)P(S)\right) = \Omega\left(h(n)\right))$$

holds.

This completes our methodology for determining the expected routing length $\mathbb{E}(R^{\mathbf{R}}_{s,e})$ for non-greedy embeddings assuming only local inaccuracies. We have applied the presented methodology for evaluating the Freenet algorithm and its alternative version, *NextBestOnce* [135, 136].

Our second scenario assumes that the monotonicity of the stochastic process $(X_i)_{i\in\mathbb{N}_0}$ cannot be guaranteed at any point during the routing. Here, we replace the process $(X_i)_{i\in\mathbb{N}_0}$ by the process $(Y_i)_{i\in\mathbb{N}_0}$ with $Y_i = \min\{X_0, \ldots, X_i\}$. $(Y_i)_{i\in\mathbb{N}_0}$ is a decreasing process and the existing results for such processes can be used. Figure 5.3c illustrates the relation of corresponding realizations of the processes $(X_i)_{i\in\mathbb{N}_0}$ and $(Y_i)_{i\in\mathbb{N}_0}$.

However, the characterization of $(Y_i)_{i\in\mathbb{N}_0}$ is drastically impeded. Recall that the probability distribution of the state X_i, the distance of the i-th node v_i to the destination e, is determined based on the state X_{i-1}, the distance of predecessor on the route, which is a neighbor of v_i. Y_i does not necessarily reveal the distance of v_i to e and Y_{i-1} is not always the distance of a neighbor of v_i. This complicates the derivation, because we cannot determine the relation between Y_{i-1} and Y_i based on a model of distances of neighboring nodes. Nevertheless, knowledge of the routing algorithm and the overlay topology generally enables to determine upper and lower bounds on X_{i-1} in terms of Y_{i-1}. Based on these bounds,

we derive the probability distribution of X_i and thus Y_i from X_{i-1}. However, the derivation and its accurateness depends on the embedding and the routing algorithm. It is hard to offer general guidelines on how to relate the two processes $(X_i)_{i\in\mathbb{N}_0}$ and $(Y_i)_{i\in\mathbb{N}_0}$.

We applied the above methodology for determining the advantage of Neighbor-of-neighbor routing in [138] and for the routing length in tree-based greedy embeddings using an alternative non-isometric distance function in Theorem 8.4. In addition, Theorems 6.3 and 6.6, which establish that virtual topologies cannot be stabilized efficiently, make use of a variant of the above methodology. Outside the area of F2F overlays, we adapted the proposed methodology to provide tight bounds on the performance of dynamic Kademlia-type system [133, 141], which are widely used large-scale discovery services integrated e.g., in BitTorrent [1] or eMule [3].

Up to now, we did not consider parallel routing algorithm. Indeed, we encounter parallel routing at multiple points throughout the thesis. We obtain the routing complexity with multiplication of the routing length of one discovered route with the number of such routes.

This completes our description of the methodology with regard to the computation complexity of routing algorithms. Our evaluation of stabilization algorithms and their complexity is similar, as detailed in the following.

5.2.2 Communication Complexity of Stabilization Algorithms

In this section, we consider the evaluation of stabilization algorithms $\mathbf{S}$. Stabilization algorithms are distributed algorithms that keep the local state of nodes in a distributed system uptodate. There are two principal ideas on how to evaluate stabilization algorithm:

1. The communication complexity resulting from a topology change, i.e., a node join or departure under the assumption that the local state before the topology change was correct.

2. The communication complexity of restoring correct local information starting from an arbitrary state information (with certain restrictions such as having a connected overlay).

In this thesis, we focus on the first approach because we are interested in the average communication complexity of the algorithm over an extended period of time. A topology change should in general only result in a small derivation from a desired state and thus require much less communication complexity than the worst-case scenario assumed by the second approach.

As in Section 5.2.1, we start with a short classification of stabilization algorithms and an overview of evaluation metrics. Then, we summarize common evaluation methods and formalize these implicitly applied methods.

Algorithm Classification: We classify stabilization algorithms in reactive and periodic algorithms. Reactive algorithms react to observed changes in the local state such as a lost connection to a neighbor. In contrast, periodic algorithms regularly update their local state. While the communication complexity of periodic algorithms is independent of the churn rate, the communication complexity of reactive algorithms is closely related to the frequency of topology changes. Furthermore, stabilization can combine periodic and reactive algorithms: a periodic algorithm $\mathbf{S}_1$ checks for changes in the neighborhood of each node and activates a reactive algorithm $\mathbf{S}_2$ if it detects a change.

We have introduced examples for stabilization algorithms in F2F overlays in Section 3. Usually, a subroutine of the stabilization consists of periodic heart-beat messages to neighbors in order to detect node departures. In addition, reactive or periodic updates of the local state are applied. For instance, virtual overlays establish and repair tunnels after topology changes. In contrast, Freenet utilizes a periodic update mechanism, namely periodic attempts to swap coordinates. However, our survey of the state-of-the-art revealed that Freenet does not fulfill our requirements. Both our candidate solutions virtual overlays and greedy embeddings rely primary on reactive algorithms. Disregarding the constant negligible overhead of heart-beat messages, our evaluation thus focuses on reactive algorithms.

Evaluation Metrics: We aim to derive the *expected stabilization complexity* $CS^{\mathbf{S}}$, i.e., the expected number of messages required by the reactive stabilization algorithm $\mathbf{S}$ after a topology change. We define the computation complexity of $\mathbf{S}$ in terms of system models, following the notation in Section 5.1.2. As $\mathbf{S}$ reacts to a change in the system model corresponding to a topology change by adapting node properties, we first need to express $\mathbf{S}$ as a transition from one system into another. Afterwards, we can define $CS^{\mathbf{S}}$ as a metric.

For this purpose, we model a topology change and the subsequent application of the stabilization algorithm as follows. First, the topology change $Q : \mathbb{SYS}_n^{\mathbf{X}_1,\dots,\mathbf{X}_r} \to \mathbb{SYS}_{n(Q)}^{\mathbf{X}_1,\dots,\mathbf{X}_r}$ with $n(Q) \in \{n-1, n+1\}$

is a random variable, which either adds or removes a node and its adjacent edges from the graph G. Second, the stabilization algorithm $\mathbf{S} : \mathbb{SYS}^{\mathbf{X}_1,\ldots,\mathbf{X}_r}_{n(Q)} \to \mathbb{SYS}^{\mathbf{X}_1,\ldots,\mathbf{X}_r}_{n(Q)}$ adapts the properties to adhere to the topology change. So, a system **Sys** is modified to a system **Sys**$'$, differing in one node and in the assignment of node properties.

Now, we define the expected communication complexity $CS^{\mathbf{S}}$ analogously to the routing complexity. Consider a realization q of the topology change Q. Let $CS^{\mathbf{S}}_q$ be number of messages $\mathbf{S}$ needs to react to q. So, the expected stabilization complexity $\mathbb{E}(CS^{\mathbf{S}})$ is the average over all realizations q, i.e.,

$$\mathbb{E}(CS^{\mathbf{S}}) = \sum_{q \in Top} \mathbb{E}(CS^{\mathbf{S}}_q) P(Q = q) \tag{5.9}$$

with Top denoting the set of all possible node additions or removals.

State-of-the-Art Evaluation: There is a vast amount of related work concerning the second evaluation method of restoring a desired state from any arbitrarily out-dated local information. The existing work considers both the methodology of (self-)stabilization in general [69] and applications of these methods to specific overlay topology, such as Chord-like rings [88, 146], skip lists or skip graphs [83], Delauney graphs [84] and spanning trees [26]. However, their method is based upon the assumption that the state information is arbitrarily out-dated while we focus on the need of stabilization after one topology change.

The first method of evaluating the communication complexity produced by individual topology changes is used implicitly when analyzing distributed systems. For instance, [154] and [97] ascertain that the stabilization complexity is polylog in Chord and Viceroy, respectively. Because their stabilization algorithm is not the fundamental contribution but rather a part of the complete system analysis, the theoretical evaluation is informal and short. We identify common strategies in these works as well as ours. In the following, we formalize these common strategies in the form of a widely applicable methodology.

Formalization of Evaluation: We here formalize one method to derive asymptotic bounds on the stabilization complexity. The approach is very generic as most parts of the evaluation depend primarily on the specific algorithm.

For a reactive stabilization algorithm, the stabilization algorithm calls different subroutines for node joins and node departures. Thus, we consider joins and departures individually. Formally, let $Top_J \subset Top$ and $Top_D \subset Top$ denote the set of possibly joins and departures, respectively. We determine the expected stabilization complexity $CS^{\mathbf{S}}$, defined in Equation 5.9, of $\mathbf{S}$ by

$$\mathbb{E}(CS^{\mathbf{S}}) = \sum_{q \in Top_J} \mathbb{E}(CS^{\mathbf{S}}_q) P(Q = q) + \sum_{q \in Top_D} \mathbb{E}(CS^{\mathbf{S}}_q) P(Q = q).$$

Now, we consider how to derive $\mathbb{E}(CS^{\mathbf{S}}_q)$. If the stabilization algorithm $\mathbf{S}$ can be modeled as a round-based protocol, we derive the communication overhead by considering the number of messages sent in each round. So, let $Z_q(j)$ be the messages sent in the j-th round. Then, we have

$$CS^{\mathbf{S}}_q = \sum_{j=1}^{\infty} Z_q(j). \tag{5.10}$$

and the expected value $\mathbb{E}\left(CS^{\mathbf{S}}_q\right)$ is the sum of expected values $\mathbb{E}\left(Z_q(j)\right)$. Figure 5.4 gives an example for stabilization after a node join in three rounds. A common possibility to derive the probability distribution of Z_j is iteratively, i.e,

$$\begin{aligned} P(Z_q(j) = i_j) = \sum_{i_1,\ldots,i_{j-1}} & P(Z_q(j) = i_j | Z_q(j-1) = i_{j-1}, \ldots, Z_q(1) = i_1) \\ & P(Z_q(j-1) = i_{j-1} | Z_q(j-2) = i_{j-2}, \ldots, Z_q(1) = i_1) \\ & \ldots \\ & P(Z_q(1) = i_1) \end{aligned}$$

In other words, by considering the earlier sent messages, we derive how many nodes are bound to send messages in the current round. Even if $\mathbf{S}$ is not strictly round-based, one can often (but not always) simplify the protocol by assuming that a node at hop distance h from the joined or departed node participates in the protocol in the j-th round. As stated above, Z_j and hence $CS^{\mathbf{S}}_q$ depend on the actual stabilization algorithm $\mathbf{S}$ as well as the underlying system model.

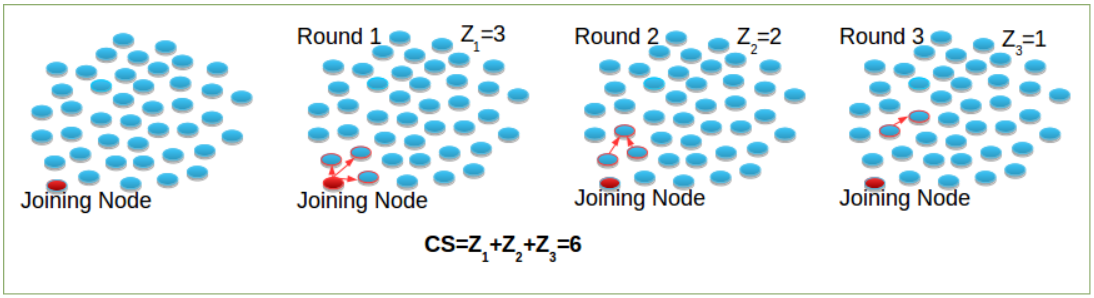


Figure 5.4: Exemplary method for determining the complexity of a stabilization algorithm: Divide the stabilization process into rounds and determine the number of messages Z_j for round j. The total complexity CS is derived as the sum over all rounds.

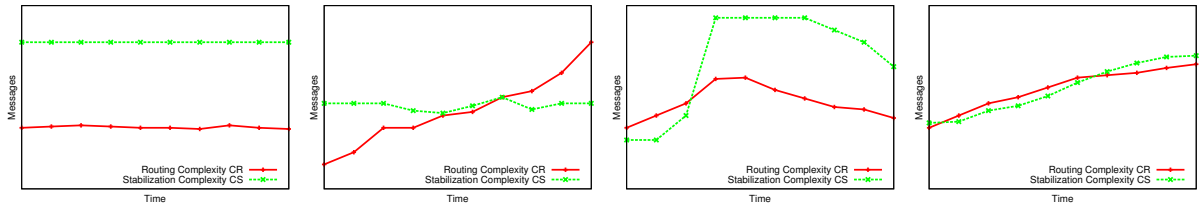


Figure 5.5: Examples of routing and stabilization complexity over time

We utilize the above method in Chapter 8 and as part of our combined evaluation of routing and stabilization complexity in virtual overlays in Chapter 6. However, the latter requires additional consideration.

5.2.3 Routing and Stabilization over Time

Up to now, we have considered routing and stabilization complexity independently of each other. However, the overall communication complexity depends on both aspects. Furthermore, they are usually not independent. In the following, we consider routing and stabilization complexity as two correlated metrics.

For that purpose, we represent our F2F overlay over time as system models $(\mathbf{SM}_t)_{t\in\mathbb{N}_0}$, so that $\mathbf{Sys}_t$ represents the system after the t-th topology change and subsequent stabilization. Topology changes are defined by the stochastic process $(Q_t)_{t\in\mathbb{N}_0}$. Following the above model, we model routing and stabilization complexity as two stochastic processes $(CR_t^{\mathbf{R}})_{t\in\mathbb{N}_0}$ and $(CS_t^{\mathbf{S}})_{t\in\mathbb{N}_0}$ such that

- $CR_t^{\mathbf{R}}$ is the routing complexity of routing algorithm $\mathbf{R}$ at time t, and
- $CS_t^{\mathbf{S}}$ is the stabilization overhead of algorithm $\mathbf{S}$ at time t.

Figure 5.5 displays several exemplary realization of $(CR_t^{\mathbf{R}})_{t\in\mathbb{N}_0}$ and $(CS_t^{\mathbf{S}})_{t\in\mathbb{N}_0}$. Usually, the two processes are correlated.

Now, in order to derive the evolution of the routing and stabilization complexity over time, we first characterize the system model $\mathbf{SM}_0$. Afterwards, we characterize the modified system models $\mathbf{SM}_t$ based on $\mathbf{SM}_0, \ldots, \mathbf{SM}_{t-1}$, the stabilization algorithm $\mathbf{S}$, and the topology change Q_t. We can then derive CR_t and CS_t for the system model $\mathbf{SM}_t$ as described in Section 5.2.1 and Section 5.2.2.

The methodology has been of particular importance for Chapter 6. We assume that the stabilization complexity in a virtual overlay are bound polylog and show that the routing length then cannot remain polylog over time under this assumption.

5.2.4 Simulation

In order to validate and concretize our asymptotic bounds, we perform simulation studies on real-world graphs. As for the theoretical analysis, we evaluate the following: i) routing length and complexity in a static system, ii) stabilization complexity in a static system, and iii) long-term trade-off of the routing and stabilization complexity in a dynamic system. In the following, we first describe our general simulation model and then focus on the individual evaluation methods. The simulation consists of sampling the processes described in previous sections.

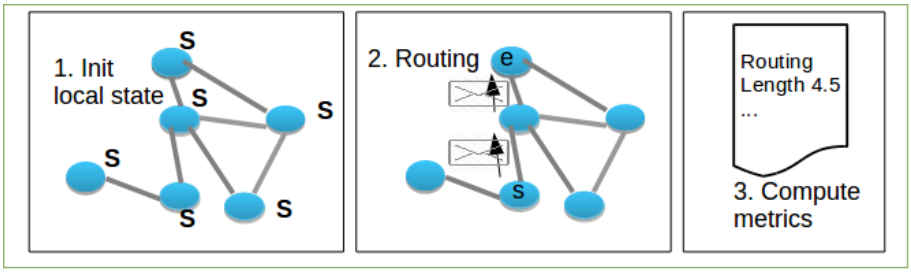


Figure 5.6: Simulation-based evaluation of the routing complexity: First initialize the system using the stabilization algorithm **S**, then execute the routing algorithm $\mathbf{R}_{node}$ for source-destination pairs (s, e), and compute the desired evaluation metrics.

General Simulation Model: We make use of the graph analysis framework GTNA, the Graph-Theoretic Network Analyzer [144]. GTNA enables generating realizations of a system model **SM** and sampling our desired metrics with respect to those realizations. For this purpose, GTNA relies on two functionalities. First, *transformations* $tf : \mathbb{SYS} \to \mathbb{SYS}$ with $\mathbb{SYS}$ denoting the set of all graph-properties pairs enable topology and property changes. In particular, a transformation might add a property such as an embedding to the graph's properties. Second, metrics $met : \mathbb{SYS} \to \mathbb{R}^*$ measure (possibly multi-dimensional) quantities about a graph and its properties. There are solely graph-dependent metrics, such as the degree distribution, and graph-property dependent metrics, such as the routing length, which depends upon the graph, the stabilization algorithm, and the routing algorithm. Our evaluation is concerned with graph-property dependent metrics.

During the simulation, we initialize a graph by reading the graph $G = (V, E)$ from a file. Afterwards, we add an initial set of properties in order to represent a F2F overlay in a valid state. Starting from this initial state, we sample metrics and execute further transformations if necessary. In order to achieve statistical significance, we repeat this process and always present our results with 95% confidence intervals.

Routing: We sample the distribution of both the routing complexity $CR^{\mathbf{R}}$ and the routing length $R^{\mathbf{R}}$ of a routing algorithm **R** for a previously initialized system. In particular, we thus obtain estimates of the expected routing complexity (Equation 5.5) and the expected routing length (Equation 5.6). For each sampled system, we execute the routing algorithm for m randomly selected source-destination pairs $(s, e) \in V \times V$ with $s \neq e$ and compute the desired values. Figure 5.6 illustrated the steps of the simulation.

Stabilization: Similarly, we sampled the distributions of the stabilization overhead by performing m topology changes on the same initial system. We considered joins and departures separately. For sampling the overhead of departures, we thus execute the following protocol

1. Remove v from the graph $G = (V, E)$
2. Execute stabilization algorithm **S**
3. Return G to its previous state for the next iteration

Similarly, the overhead of node joins is sampled by adding a new node as well as edges to existing nodes according to some random variable, executing the stabilization algorithm, and returning the graph to its initial state.

Routing and Stabilization over Time: Our time-dependent analysis first specifies the order of topology changes as a realization of the stochastic process $(Q_t)_{t \in \mathbb{N}_0}$ from Section 5.2.3 for its first t_{MAX} steps. Afterwards, we execute t_{MAX} steps consisting of i) topology change, and ii) corresponding stabilization. In each step, we thus sample the stabilization complexity CS_t. In addition, we sample the routing complexity CR_t every s_r steps.

The stochastic process $(Q_t)_{t \in \mathbb{N}_0}$ is derived from the churn patterns measured in Freenet and introduced in Section 4.1.3. We use both the original traces and the derived models.

Our analysis of a system over time does not consider that topology changes can interrupt the stabilization or the routing. Whereas the design of our algorithms should certainly treat such events adequately, we show that the theoretical bounds are representative despite the simplification to non-interfering topology changes.

5.3 Effect of Churn

One of the limitations of the model presented in Section 5.2.3 is the assumption that each topology change is repaired before the next. In practice, stabilization of several changes generally takes place in parallel. Thus, they potentially affect each other. The number of messages required for stabilization after two inter-dependent changes is not necessarily given by the sum of the messages required to deal with each change subsequently. We are thus interested in the effect of churn on routing and stabilization. More specifically, we consider the probability that routing or stabilization are affected by further topology changes. By showing that this probability is usually small, we reason that our simplified analysis of a dynamic F2F overlay as a sequence of topology changes with immediate stabilization is reasonably realistic.

We start by introducing some related work on how churn models can be integrated into the theoretical analysis of overlays in Section 5.3.1. In Section 5.3.2, we specify the methodology for determining the probability that an action such as routing is affected by a topology change in terms of the session length distribution S. Afterwards, in Section 5.3.3, we detail how to derive an upper bound on the desired probability. Last, in Section 5.3.4, we evaluate our methodology based on the churn models for Freenet and find the probability that routing or stabilization is affected by a topology change is small. The concepts introduced in this section and their evaluation have been partially published in [131].

5.3.1 State-of-the-Art

Few theoretical works consider the impact of churn on properties of distributed systems due to the complexity of most scenarios. The existing approaches focus on the impact of the session length distribution on the probability that a certain set of nodes, which is required to provide a service, fails. However, they all focus on overlay connectivity and the lifetime of content within the overlay. Connectivity, for instance, is treated by Leonard et al. [93]: They express the time until a node does not have any neighbors in terms of the session length and identify overlay topologies with a low probability of fragmentation. In [39], it is shown that a lifetime-based neighbor selection reduces the number of failing links under the assumption of a Pareto distributed session length. The importance of the session length distribution on lifetime of content, i.e., the time for which inserted content is stored by at least one online node, is shown in [164]. The authors prove that an exponential session length requires that files are periodically re-inserted, whereas a heavy-tail distribution does not. Yao et al. model content lifetime in the light of heterogeneous session lengths, allowing a more fine-grained prediction of future behavior. They improve the availability through enhanced storage schemes [11]. In contrast to the related work, we target a more complex scenario, because routing and stabilization depends on latencies and processing times in addition to the session length.

The impact of churn on routing is only considered in simulation and measurement studies. The first are of limited scalability, often only a few thousand nodes are considered. The latter are restricted to deployed systems and hence do not allow the evaluation of newly designed algorithms. Our method offers a more scalable alternative to simulations and can be applied in combination with simulations, such that the two evaluation techniques validate each other to reduce implementation and design errors.

5.3.2 General Methodology

In this section, we show how to derive the probability that a node departure affects an action such as routing and stabilization. The probability that a node join affects an action can be computed in a similar fashion. However, node joins generally have a less drastic effect as they do not destroy existing routes.

The main component of our model is the session length S, i.e., the time between a node's join and its departure. So, the sessions of users are assumed to be independent and identically distributed. Furthermore, we assume that the session length S i) has a continuous distribution function F_S, i.e., the density f_S exists, and is integrable with expected value $\mathbb{E}(S)$.

For the derivation, let A be an action, e.g., routing or stabilization after a topology change, and denote by $P(A)$ the probability that A is unaffected by any node departures. Let $v_1, \ldots, v_m$ be the nodes required for action A, and let D_i and R_i denote the time v_i is last required and the remaining time of v_i's session, respectively. Because the action A remains unaffected if all nodes $v_1, \ldots, v_m$ are still online at the time they are last required, we get

$$P(A) = P\left(\bigcup_{i=1}^{m} R_i \geq D_i\right). \tag{5.11}$$

While D_i is defined by application-specific parameters, the remaining online times R_i are all identically distributed and their common distribution R can be determined from the session length S. In the following, we thus derive the distribution of the remaining online R of a contacted node.

Now, we determine R in terms of f_S, F_S, and $\mathbb{E}(S)$. First, we consider the session length distribution C of a contacted online node. In other words, we are interested in the session length of a node given that it is contacted. In this context, $L_x(contacted)$ denotes the likelihood of a node with session length x to be contacted. Various likelihood functions are possible.We determine the density of C by

$$f_C(x) = \frac{L_x(contacted) f_S(x)}{\int_0^\infty L_y(contacted) f_S(y) dy}. \tag{5.12}$$

We now show how to obtain the distribution of R for an arbitrary likelihood function before considering the case that the online time of a node and its likelihood to be involved in the action are independent. The time at which a node is contacted is assumed to be uniformly distributed within its session. Hence, the remaining online time of the contacted node is then $R = U \cdot C$ distributed for a uniform distribution U with values in $[0, 1]$. With the normalization constant $N_S = \int_0^\infty L_y(contacted) f_S(y) dy$, the density of R is

$$\begin{aligned} f_R(x) &= \int_0^\infty \frac{1}{y} f_U(y) f_C\left(\frac{x}{y}\right) dy \\ &= \frac{1}{N_S} \int_0^\infty \frac{1}{y} f_U(y) L_{x/y}(contacted) f_S\left(\frac{x}{y}\right) dy \\ &= \frac{1}{N_S} \int_0^1 1/y L_{x/y}(contacted) f_S\left(\frac{x}{y}\right) dy. \end{aligned} \tag{5.13}$$

We now determine Equation 5.13 under the assumption that the online time of a node and its likelihood to be involved in the action are independent. The assumption is reasonable for F2F overlays due to the independence of the degree of the online time, so that in contrast to other P2P systems nodes do not gain influence by increasing their number of neighbors over time. Note that the assumption can also be seen a worst-case assumption in adherence to the fact that nodes that have been online longer are less likely to leave the system and algorithms usually tend to prefer long-lived nodes rather than short-lived. We present more complex scenarios in [131] but do not require them in this context.

If a node is selected independently of the time it is already participating, a contacted node is selected uniformly at random from all *online* nodes. Note that the probability of a node to be one of the currently online nodes increases linearly with its session length. In other words, given the set of all sessions, we select a session with a probability proportional to its length. We illustrate the relation between the session length and the probability of contact in Figure 5.7. As a consequence, the likelihood function is given by $L_x^{ind}(contacted) = x$, and thus the remaining online R^{ind} has the density function

$$\begin{aligned} f_{R^{ind}}(x) &= \frac{1}{\int_0^\infty y f_S(y) dy} \int_0^1 1/y \frac{x}{y} f_S\left(\frac{x}{y}\right) dy \\ &= \frac{1}{\mathbb{E}(S)} \int_0^1 -\left(\frac{x}{y}\right)' f_S\left(\frac{x}{y}\right) \\ &= \frac{1}{\mathbb{E}(S)} \left(\lim_{y \to 0} F_S(x/y) - F_S(x)\right) \\ &= \frac{1 - F_S(x)}{\mathbb{E}(S)} \end{aligned} \tag{5.14}$$

with g' denoting the derivative of a function g [1]. In Figure 5.8, we illustrate the relation between the session length S, the session length of online nodes C, and the remaining online $R = R^{ind}$ for a log-normal distributed session length.

5.3.3 Worst-Case Bounds

Now, we consider worst-case bounds on the probability $P(A)$ as defined in Equation 5.11. For m nodes, the worst possible scenario, i.e., the one for which they are needed longest, is that the nodes are contacted subsequently and a reply from the last node is returned to the initial node along the same path. In other words, we evaluate the probability that forwarding along the path $(v_1, v_2, \ldots v_{m-1}, v_m, v_{m-1}, \ldots, v_1)$ succeeds. In the terminology of Section 5.2.1, our worst-case analysis considers a single-threaded routing

[1] [11] presents an alternative formal derivation of Equation 5.14 for heterogeneous session length distributions.

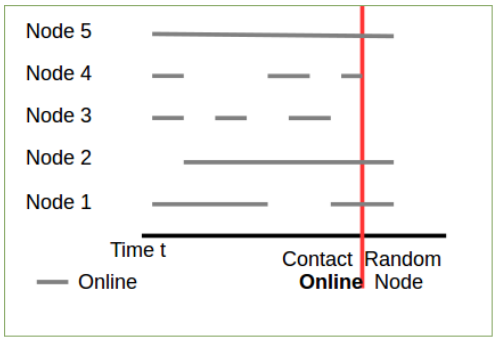


Figure 5.7: The session length of randomly contacted *online* node is usually longer than average session length because the probability of a session to be selected scales linearly with its length.

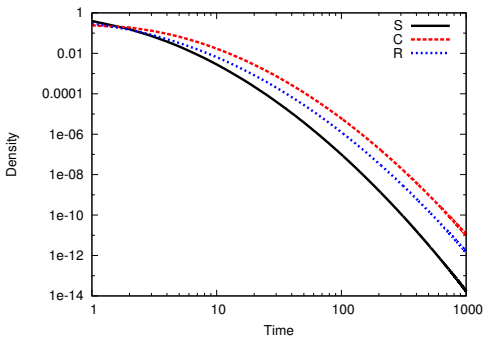


Figure 5.8: The probability densities of i) the session length S, ii) the session length of an online node C, and iii) the remaining online time R for a standard log-normal distribution ($k = 1$, $\lambda = 1$, see Table 5.1)

algorithm. In the following, we determine the probability of a topology change to affect single-threaded routing in terms of the number of messages $2m$. The results present an upper bound on the probability of a topology change to affect an action requiring $2m$ messages.

In order to apply Equation 5.11, we need to determine D_i, the time a node v_i is required to remain online. Here , we consider two cases:

1. *Predefined path selection*: The routing depends on a certain path. Hence, the routing fails if v_i leaves the system, regardless if its departure is detected by v_{i-1}.

2. *Free path selection:* The routing does not require a certain path, so that nodes on the path can always forward the request if they have at least one online neighbor. If v_i detects that a neighbor u has departed the system, v_i does not forward to u. Thus, we know that v_i considered v_{i+1} to be online when forwarding the request.

Therefore, predefined path selection usually requires a shorter remaining online time for the nodes.

F2F overlay routing and stabilization algorithms can rely on predefined (but implicit) path selection or free path selection. Greedy routing, which fails if a node has no closer neighbor, is a combination of both classes. Nodes might chose alternative neighbors if several closer neighbors exists but the routing length is likely to be different. Thus, predefined path selection is a lower bound on the probability that greedy routing succeeds. In contrast, random walks as used for the Freenet stabilization algorithm [48] are a typical example of free path selection. We show how to derive bounds for both scenarios.

Before the actual derivation, we introduce some additional notation. Let LAT and RTT be the latency and round-trip time distribution in the network, T the time to transmit a message, i.e., the time between sending the first and the last bit of the message, P the processing time of a message when selecting the next node, and F be the processing time when forwarding the reply. Denote the latency between v_{i-1} and v_i by $LAT_{i-1,i} \sim LAT$ and the round trip time of v_{i-1} and v_i by $RTT_i = LAT_{i-1,i} + LAT_{i,i-1} \sim RTT$, the transmission time between node v_i and v_j by $T_{ij} \sim T$, and the processing time for finding the next hop and forwarding at v_i by $P_i \sim P$ and $F_i \sim F$. Abbreviate $T_i = T_{(i-1)i} + T_{i(i-1)}$ and set $P_m = 0$ because the last node does not require selecting a new successor. A graphic interpretation of the various delays is displayed in Figure 5.9.

Lemma 5.6. *If the path selection is predefined, the node v_i's remaining online time has to exceed*

$$D_i = \begin{cases} \sum_{j=2}^{m} (LAT_{j-1,j} + T_{j-1,j} + P_j) + F_m + T_{m,m-1}, & i = m \\ D_{i+1} + F_i + LAT_{i+1,i} + T_{i,i-1}, & 1 < i < m \\ D_2 + LAT_{2,1} \end{cases} . \tag{5.15}$$

Proof. Note that all nodes need to remain online until the request reaches v_m. The i-th step on this path, i.e., the transition from v_i to v_{i+1}, requires the processing time P_i, the transmission time $T_{i,i+1}$, and the latency $LAT_{i,i+1}$. As a consequence, the endpoint v_m on the path has to remain online for the time $\sum_{j=2}^{m} (LAT_{j-1,j} + T_{j-1,j} + P_j)$ as well as its forwarding and transmission time F_m and $T_{m,m-1}$. Afterwards, v_m is no longer required for the routing to be successful. Hence, the first case in Equation 5.15 holds. After the node v_{i+1} has forwarded the reply, the node v_i has to stay online until it receives the message from v_{i+1}, meaning the time $LAT_{i+1,i}$ in addition to the time D_{i+1} that v_{i+1} is required.

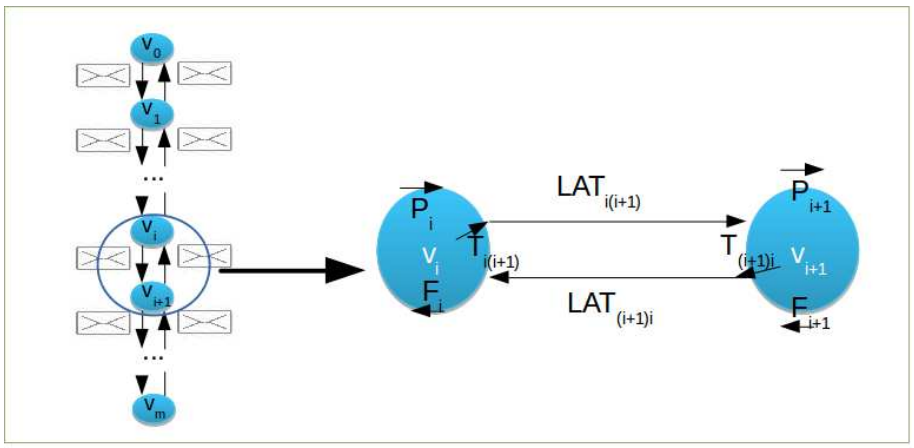


Figure 5.9: Single-threaded recursive routing with m nodes (left) and the delays influencing the successful termination (right): Latencies $LAT_{i(i+1)}$, transmission delays $T_{i(i+1)}$, processing times P_i and F_i for request and reply

For the second case, v_i then has to process and forward the reply, thus adding transmission time $T_{i,i-1}$ and the forwarding time F_i. The initiator of the routing does not forward and transmit the reply, thus the third case follows. □

If the path selection is free, nodes can change the path when detecting a neighbor's departure, so that the manner of detection is essential for D_i. Here, we assume a controlled leave, meaning that nodes either inform their neighbors before departing or that the loss of synchronous connections are detected by said neighbors. As a consequence, the expired time between a node's departure and its neighbor's acknowledgment of the departure is given by the latency between the two nodes. Then, we get the following result for the time D_i.

Lemma 5.7. *If the path selection is free, the node v_i's remaining online time has to exceed D_i, recursively defined by*

$$D_i = \begin{cases} RTT_i + T_i + F_i + P_i, & i = m \\ D_{i+1} + RTT_i + T_i + F_i + P_i, & i = 2 \dots m-1 \\ \sum_{j=1}^{m} D_j, & i = 1. \end{cases} . \tag{5.16}$$

Proof. For showing that indeed Equation 5.16 holds, first consider the last node v_m on the path. Due to the controlled leave, v_{m-1} knows at time t that v_m has been online at time $t - LAT_{m,m-1}$. The time v_m then has to remain online starting at time $t - LAT_{m,m-1}$ is given by the sum of the following

- $LAT_{m,m-1}$: time from moment v_m is last known to be online until message transmission starts
- $T_{(m-1)m}$: v_{m-1} transmits the request
- $LAT_{m-1,m}$: v_m receives request
- $P_m + F_m$: v_m processes
- $T_{m(m-1)}$: v_m transmits the reply

So, the first case follows.

With exception of the initiator v_0, the time D_i for which v_i is the required to remain online since the time it has last known to be online consists of the following

- $LAT_{i(i-1)}$: time from moment v_i is last known to be online until message transmission starts
- $T_{(i-1)i}$: v_{i-1} transmits request
- $LAT_{m-1,m}$: v_i receives request
- P_i: v_i processes request
- $D_{i+1} - L_{i+1,i}$: time v_{i+1} has to remain online after v_i starts transmitting the message
- $L_{i+1,i}$: v_i receives reply from v_{i+1}

$\mathbf{S}$	$W(\lambda, k)$	$ln\mathcal{N}(\lambda, k)$
$\mathbf{f_S(x)}$	$k\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$	$\frac{k}{\sqrt{2\pi}x} e^{-k^2 \ln^2(x/\lambda)/2}$
$\mathbf{P(S \geq z)}$	$\frac{e^{-(x/\lambda)^k}}{\lambda\Gamma(1+1/k)}$	$\Phi\left(k \ln \frac{z}{\lambda}\right)$
$\mathbf{f_{R^{ind}}(x)}$	$\frac{e^{-(x/\lambda)^k}}{\lambda\Gamma(1+1/k)}$	$\frac{1-\Phi(k \ln(x/\lambda))}{\lambda e^{1/(2k^2)}}$
$\mathbf{P(R^{ind} \geq z)}$	$1 - \frac{ze^{-(z/\lambda)^k} + \lambda\gamma(1+1/k, \left(\frac{z}{\lambda}\right)^k)}{\lambda\Gamma(1+1/k)}$	$1 - \frac{z(1-\Phi(k(\ln z/\lambda))) + \lambda e^{1/(2k^2)}\Phi(k \ln(z/\lambda) - 1/k)}{\lambda e^{1/(2k^2)}}$

Table 5.1: Remaining online time R^{ind} for common session length distributions. $\Gamma(k) = \int_0^\infty z^{k-1}e^{-z}dz$ and $\gamma(k,x) = \int_0^x z^{k-1}e^{-z}dz$ denote the Gamma function and the incomplete Gamma function, respectively; $\Phi(x) = \int_{-\infty}^x \sqrt{2\pi}e^{x^2/2}$ the distribution function of the standard normal distribution.

- F_i: v_i processes the reply
- $T_{i(i-1)}$: v_{i-1} transmits reply

The second case in Equation 5.16 follows as the sum of the above times. The third case holds as the initiator does not require further processing and communication with a predecessor on the path. □

The probability that a departing node affects single-threaded routing, defined in Equation 5.11, can now be determined based on Lemmata 5.6 and 5.7. However, an exact derivation is only possible for very simple scenarios such as constant latencies and processing times. Usually, we rely on Monte-Carlo sampling, sampling n realizations $d_0, \ldots, d_n$ of $(D_0, \ldots, D_m)$ based on the distributions LAT, RTT, T, P, and F. For the j-th sample $d_j = (d_{j0}, \ldots, d_{jm})$, we compute the success probability $s_j = \prod_{i=0}^m P(R > d_{j0})$. The overall success probability is then approximated by $s = 1/n \sum_{j=1}^n s_j$. In the following, we evaluate the probability for real-world data.

5.3.4 Evaluation

In this section, we evaluate the probability that single-threaded routing fails and thus give worst-case bounds on the probability that stabilization and routing are interrupted. First, we shortly describe our set-up and data sets, followed by our expectations and results. We conclude by interpreting the results with regard to our evaluation methods.

Equation 5.11 depends on the session length distribution S as well as the distributions LAT, RTT, T, P, and F of the different delays, as defined in Section 5.3.3. For simplicity, we disregard the transmission time T and the local computation time P and F because those are usually negligible in comparison to the latency. Then, we make use of the data sets described in Section 4.1.3 for the session length S, i.e., we consider Weibull and Lognormal distributed session length with parameters fitted to the Freenet traces. Furthermore, we assume symmetric latencies for simplicity and utilize the IPlane data set modeling Internet latencies [10].

Now, Equation 5.11 requires the computation of the remaining online time R from these session lengths. We assume that the probability of contacting a node is independent of its online time. In practice, nodes with longer online times are usually more likely to be contacted. As they are also more likely to have a longer remaining online, the assumption presents a worst-case scenario. We derive the desired distribution based on Equation 5.14 using basic integration techniques. Due to their limited importance to the purpose of this section, we provide the details of the integration in Appendix C. The resulting distributions are displayed in Table 5.1.

We expect an exponential decrease in the probability of successfully completing the routing with the number of involved nodes m. In order to see the reason for the exponential decrease, let A_m denote single-threaded routing with $m+1$ nodes. For A_1 to be successful, the one involved node has to remain online at most as long as any node for A_m, so that indeed

$$P(A_m) \leq P(A_1)^m.$$

However, since sessions in Freenet are often in the order of hours or at least minutes whereas an action is completed within seconds, the decrease should be slow in the beginning, achieving only negligible failure rates for up to several tens of nodes.

Our expectation is verified by the results. Figure 5.10 displays the probability of the routing to be unaffected by the topology change. For up to 100 nodes, the probability that one of them leaves while required to complete an action is less than 2% for predefined routes, and only about 1% for free routes. We will see that for efficient F2F overlays, the number of nodes involved in an action is usually below

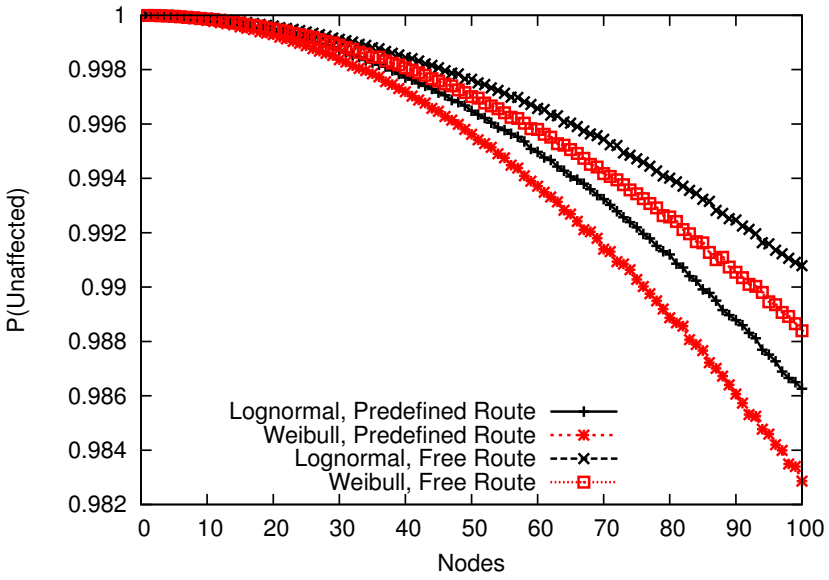


Figure 5.10: Probability of an action to remain unaffected by further topology changes in terms of number of involved nodes (Upper bound)

100. Thus, the results of this worst-case analysis validated that interfering topology changes are rare and can be overlooked for the asymptotic bounds.

5.4 Robustness and Censorship-Resistance

Despite the low probability for interruptions to routing on short routes, such interruptions still occur in practice. In addition, nodes might not forward requests due to congestion or malicious intent. Thus, the routing algorithm is required to deal with such interruptions in an adequate manner, either by finding an alternative route or declaring the routing failed. The quality of an algorithm to provide a correct result despite the existence of failures or attacks, is called *disruption tolerance*. We focus on two aspects of disruption tolerance, namely robustness and censorship-resistance as defined in Section 2.6. In the following, we start by introducing the common methodology for both aspects, based upon the methodology presented in [153] for network resilience in general. Afterwards, we discuss the methodology specific to robustness and censorship-resistance.

For brevity, we sometimes summarize robustness and censorship-resistance under the term resilience, though resilience is a more general topic.

5.4.1 General Approach

Note that we only consider the property of a system to provide the desired service despite failures or attacks rather than reacting to them. In other words, we focus on proactive rather than reactive failure and attack mitigation. We partially consider the reaction to (node) failures by designing stabilization algorithms, whose evaluation we described in Section 5.2.2. Reacting to attacks by detecting the attackers and excluding them from the service is a complementary approach, which is out of scope for this thesis.

Disruption Classification: In general, one distinguish node and edge failures/attacks. As detailed in our attacker model in Section 2.5, we focus on internal adversaries controlling nodes. Furthermore, we disregard the routing process in the underlay. In particular, we assume that the routing protocol in the underlay deals with edge failures and concentrate on node failures. For this reason, we focus on malfunctioning or intentional misbehaving nodes rather than unreliable edges.

As motivated in Section 2.5, we focus on minimizing the impact of failures and deliberate dropping of requests on the messaging between two nodes s and e. We use multiple categories to classify the nature of service disruption:

- *Unintentional* or *Intentional*: Unintentional disruptions correspond to failures or faults in the implementation. Usually, they are uncorrelated and affect random nodes. Intentional disruptions corresponds to attacks by malicious nodes, which willfully damage the service and aim to maximize the damage.

- *All*, *Selective*, or *Random*: The affected nodes can indiscriminately sabotage all requests, intentionally sabotage selective requests, or randomly sabotage some but not all requests. The latter is common for uncorrelated failures, while selective sabotage implies intentional disruption.

- *Detectable* or *Undetectable*: We call a failure or attack detectable if nodes forwarding a request to a malfunctioning or malicious node can detect that the request was sabotaged. Otherwise, it is undetectable.

We consider both unintentional and intentional disruptions but assume the disruptions to be complete and detectable. In our scenario, complete sabotage corresponds to the dropping of any routing request. In the absence of reactive mechanisms that penalize dropping or prefer previously reliable neighbors, the impact of a misbehaving node is maximized by dropping all requests. Hence, we consider a worst-case scenario. If a system provides sufficient robustness and resistance in such a scenario, it can also withstand less powerful attackers.

The assumption of detectability is clearly valid for node failures because neighboring nodes notice the absence of heart-beat messages. If a node drops messages, the lack of a suitable response within a certain time, such as an acknowledgment signed with the private key of the receiver, enables detection. As it remains unclear which node dropped the request, the lack of a response does not reveal the actual attacker but the disruption itself is revealed. As a consequence, we assume all disruption to be detectable. Detectability is an important assumption as it allows the use of alternative routes after realizing a disruption.

Metrics: The main metric of interest for the disruption tolerance is the fraction of successfully delivered requests. Let $Succ^{\mathbf{R}}(s, e)$ be the metric that is 1 if the routing algorithm $\mathbf{R}$ successfully discovers a path from the source s to the destination e and 0 otherwise. We then define the *success ratio* $Succ^{\mathbf{R}}$ of $\mathbf{R}$ by

$$\mathbb{E}(Succ^{\mathbf{R}}) = \frac{1}{n(n-1)} \sum_{s,e \in V \setminus U, s \neq e} \mathbb{E}\left(Succ^{\mathbf{R}}(s, e)\right). \tag{5.17}$$

for a system model with n nodes and a set U of *unresponsive nodes*, i.e., either failed or malicious nodes. In addition to the success ratio, the impact of the disruption on the efficiency and scalability metrics presented in Section 5.2 are considered.

When evaluating the resilience of systems, we usually have a parameter characterizing the strength of the disruption. For instance, the number of failed nodes can be used as such a parameter. Then, we aim to derive a relation between the strength of the disruption and the success ratio (or a different metric of interest). If the success ratio drops slowly with increasing strength, the system is considered resilient.

Methodology: We evaluate the disruption tolerance both theoretically and through simulations. However, the theoretical analysis is limited, so that the focus lies on simulations. Before detailing our methodology for both, we provide a formal model of failures and attacks in the context of system models.

We model a system with failures or attacks by defining i) the group of failed or malicious nodes, and ii) the behavior of these nodes. In the following, we use the term *unresponsive* nodes to refer to either failed or malicious nodes.

We extend the notion of system models from Section 5.1.2 to include unresponsive nodes and their selection. When evaluating disruption tolerance, we assume that all systems have a property *respond*, which is 0 for unresponsive nodes and 1 for all others. An algorithm *Att* parametrized by the strength of the disruption assigns the values of *respond*. For instance, *Att* might select random failed nodes. Nodes v in a F2F overlay for which $respond(v) = 0$ adapt their behavior as follows:

1. Instead of the stabilization algorithm $\mathbf{S}$, they apply a modified algorithm $\mathbf{S}'$.

2. They drop all routing requests.

The two algorithms *Att* and $\mathbf{S}'$ determine the manner of sabotage, consisting of the selection or integration of unresponsive nodes and their manipulation of the local state. As a result, a system model $\mathbf{SM} = (\mathbf{GM}, \mathbf{PM})$ now depends on the modified stabilization algorithm $\mathbf{S}'$ and the algorithm *Att* in addition to the graph model $\mathbf{GM}$ and the stabilization algorithm $\mathbf{S}$.

Our theoretical analysis is limited to showing that we improve the disruption tolerance of a system to a certain attack strategy or failure model. More precisely, we consider two system models $\mathbf{SM}_1 = (\mathbf{GM}, \mathbf{PM}_1)$ and $\mathbf{SM}_2 = (\mathbf{GM}, \mathbf{PM}_2)$ with the same graph model $\mathbf{GM}$ and show that the expected success ratio in systems sampled from $\mathbf{SM}_2$ exceeds the success ratio for systems sampled from $\mathbf{SM}_1$. Alternatively, we show the that a routing algorithms $\mathbf{R}_1$ achieves a higher success ratio than a routing

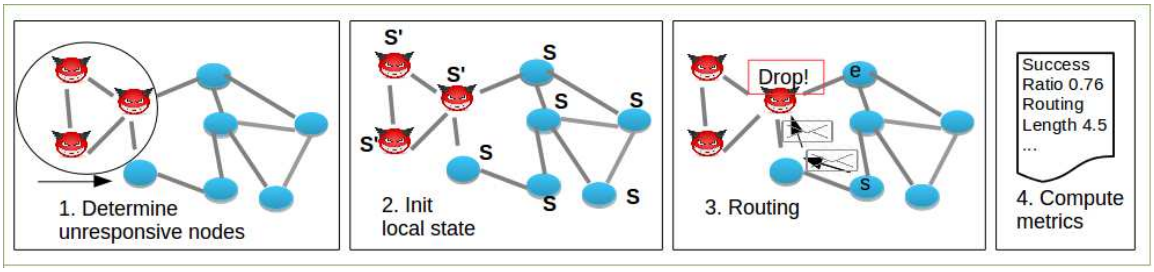


Figure 5.11: Simulation-based evaluation of disruption tolerance: First, determine faulty or malicious nodes (red), initialize the system using stabilization algorithm $\mathbf{S}$ for functional nodes (blue) and $\mathbf{S}'$ for faulty or malicious nodes, then execute the routing algorithm $\mathbf{R}_{node}$ for source-destination pairs (s, e) of functional nodes, and compute the desired evaluation metrics (Compare Figure 5.6 for the evaluation in the absence of failures and attacks)

algorithm $\mathbf{R}_2$ with regard to either the same or two distinct system models. However, we quantify neither the difference between two systems nor the actual success ratio.

The quantification of the disruption tolerance follows by simulation of failures or attacks. For this purpose, we extend the simulation model described Section 5.2.4 by adding an initial step and modifying the system initialization. So, a simulation run consists of the following steps:

1. Determine the unresponsive nodes (Algorithm $Att(x)$ for disruption strength x)

2. Initialize the system, including specific behavior S' of unresponsive nodes

3. Perform routing between pairs of *responsive* nodes, unresponsive nodes drop all messages they receive

4. Calculate metrics, in particular the success ratio

The four steps are illustrated in Figure 5.11. We evaluate different scenarios, corresponding to different strategies Att, modified stabilization algorithms $\mathbf{S}'$, and disruption strengths x. In particular, failures and attack induces different restrictions on Att, $\mathbf{S}'$, and x.

5.4.2 Robustness

In this section, we consider node failures. More precisely, we assume that nodes leave the overlay and disrupt the routing as the necessary stabilization has not yet taken place.

Thus, we define the algorithm Att and $\mathbf{S}'$ as follows. We assume node failures to be random. In other words, Att selects a fraction of nodes uniformly at random. The strength x thus is the fraction of failed nodes. As nodes only fail during the routing, they execute the same initialization as the remaining nodes, i.e., $\mathbf{S}' = \mathbf{S}$. So, the evaluation of failures is straight-forward and independent of the actual system design.

5.4.3 Censorship-Resistance

We consider an attacker that infiltrates the overlay by establishing connections to honest users. The attacker can then insert an arbitrary number of nodes in the overlay. Afterwards, the attacker modifies the initialization with the goal of reducing the success ratio by tricking many nodes to forward requests to a malicious node. During the routing, all messages are then dropped, resulting in a denial-of-service attack.

The algorithm Att consists of i) establishing connections to honest nodes, and ii) deciding of the number nodes and their connections. In the absence of suitable models for social engineering attacks, we assume that the attacker establishes edges to randomly chosen nodes. The strength x of the attack then corresponds to the number of edges the node can establish. After being integrated in the overlay, the attacker executes a modified stabilization algorithm $\mathbf{S}'$. $\mathbf{S}'$ manipulates the assignment of the local state information such a large fraction of requests are expected to fail. The nature of the manipulation depends on the routing and stabilization algorithms. We thus describe the respective behavior when evaluating a particular approach.

This completes the methodology for evaluating robustness and censorship-resistance. The evaluation method is mostly empirical: Failures or attacks are simulated to characterize their impact on the overlay. While we have a common straight-forward evaluation of failures, the behavior of the attacker highly depends on the overlay protocol.

In addition to censoring the communication between two parties, an attacker can merely wish to observe the communication and identify the communicating parties. We thus consider anonymity metrics and methods in the following.

5.5 Anonymity and Membership-Concealment

By restricting direct communication to the devices of mutually trusted users, F2F overlays seemingly provide anonymity and membership-concealment against untrusted end devices by design. However, routing and stabilization algorithms might nevertheless reveal sensitive information allowing the identification of communicating parties. Furthermore, as motivated in Section 2.6, we also require sender and receiver anonymity towards overlay neighbors.

5.5.1 Anonymity

In this section, we shortly motivate our metric used to characterize the anonymity. Furthermore, we give a short overview of anonymization in P2P systems and explain why we focus on using hop-by-hop anonymization. As stated in the introduction, this thesis' focus lies mainly on the efficiency and resilience. However, we aim to provide anonymization in order to i) prevent adversaries from detailed profiling of nodes in the system apart from their neighbors, and ii) obfuscate their neighbor's communication to some degree.

Anonymity Metrics: In the context of this thesis, anonymity is defined as unlinkability between a node and an action, such as sending or receiving a request. Here, unlinkability is defined as the inability of an attacker to decide if two or more so called *items of interest*, e.g., objects or actions, are related [121] with sufficient *certainty*. Measuring this certainty and defining which degree of certainty is sufficient has been an active area of research for decades. We focus on the most important concepts, a detailed survey can be found in [50]. A unified framework for anonymity metrics is presented by AnoA [30], which expresses anonymity in terms of indistinguishability.

There are set-based or probability-based metrics. A straight-forward concept to measure anonymity is the *anonymity set*, the set of nodes that might have sent or received a message. However, this metric is limited, as the probability to be the sender and receiver might greatly vary between the nodes in the anonymity set. As a consequence, probability-based metrics are more suitable in most scenarios, assigning each node a certain probability to be the related to the item of interest. Probability-based metrics either summarize the probability of a set of nodes in one scalar, such as Shannon's entropy [147], or consider nodes individually [159].

In the context of probability-based metrics, there are two main underlying approach for defining the probability of de-anonymization, i.e., identifying the relation between two items of interest. The first approach considers de-anonymization as a binary concept and characterizes the fraction of scenarios that allow de-anonymization. This approach is most applicable if the relation can be identified with absolute certainty in some scenarios while others do not reveal any relevant information. For instance, de-anonymization in a mix is only possible if all mixes are controlled [43], thus probability of an attacker controlling all mixes is an anonymity measure. The second approach considers a concrete scenario and assigns nodes a likelihood to be related to the item of interest. For instance, in Crowds [126], nodes forward a request to a random node with a certain probability. Based on this forwarding probability, an attacker can estimate the probability that a node forwarding the request is the actual sender. The second type of metric is applicable if the attacker gains information about the sender or receiver but is unable to derive her identity with absolute certainty. Then, anonymity is not a binary concept and more complex metrics need to be applied.

We now motivate and formally define our anonymity metric for F2F overlays. Routing in F2F overlays generally reveals some information. For example, a node known to be on the routing path might be more likely to be the sender or receiver than an arbitrary node. So, we cannot characterize anonymity as a binary concept, meaning that we focus on metrics assigning nodes a certain probability to be related to the item of interest.

Formally, we define our anonymity metric in the manner of the *degree of anonymity* in [126] by Reiter and Rubin. Let $\mathbb{SYS}_n^{M_1,\ldots,M_r}$ be a system model for a F2F overlay with a fixed topology $G = (V, E)$. Furthermore, let $v \in V$ and $\mathbf{Sys} \in \mathbb{SYS}_n^{\mathbf{X}_1,\ldots,\mathbf{X}_r}$. We then define $q_{\mathbf{Sys}}(v)$ to be the probability that v is

either the sender or receiver from the attacker's point of view. The anonymity of a node v is then the complement of the probability assigned to v, i.e., $ano_{\mathbf{Sys}}(v) = 1 - q_{\mathbf{Sys}}(v)$.

Reiter and Rubin complement their anonymity metric for a node $v \in V$ with regard to an item of interest IoI with the following concepts of *suitable certainty of not being identifiable*

- v is *beyond suspicion* if $ano_{\mathbf{Sys}}(v) \geq ano(u)$ for any node $u \in V$ in the system, i.e., v is no more likely to be related to IoI than any other node u.
- v is *probable innocent* if $ano_{\mathbf{Sys}}(v) \geq 0.5$, i.e., v is at least as likely not to be related to IoI as it is to be related.
- v is *possibly innocent* if $ano_{\mathbf{Sys}}(v) > \epsilon > 0$ for ϵ, i.e., there is non-trivial probability that v is not related to IoI

In this thesis, we only require possible innocence due to the difficulty of proving the anonymity while maintaining routing efficiency. Yet, improving the bounds on the anonymity is an important aspect of future research but out of scope for this thesis.

Methodology: Proofs of anonymity are restricted to certain adversary models. Thus, even if a correct proof of the anonymity is given, an attacker that does not confirm with the attacker model might still break the anonymity. As a consequence, defining a realistic use case and deriving a suitable attacker model is the first step in evaluating anonymity, followed by the second step of actually proving the anonymity under the previously defined model.

Three common attacker models are:

- A local internal adversary controls a certain number of participants within the network. It can thus observe and manipulate all messages it receives but cannot observe the complete system.
- A global passive adversary can observe all sent messages. However, it cannot manipulate them or observe the internal processing at a node (e.g., encrypting or decrypting a message).
- An adversary can be both global passive and local active.

David Chaum proposed two approaches for dealing with combined global passive and local active adversaries, namely Dining Cryptographers Networks [44] and Mix Networks [43]. Due to their high communication complexity or long delays, only high-latency applications such as the anonymous email service Mixminion [57] realize Chaum's ideas such that protection against a global passive adversary is achieved.

Low-latency applications, such as anonymous Web browsing realized by the distributed server solutions Tor [61] and ANON [34] as well as P2P-based solutions such as Torsk [105] or I2P [54], achieve lower delays but do not guarantee anonymity against a global passive adversary. Rather, they focus on a local internal active attacker. In the context of P2P overlays, DHTs have received particular attention with the goal of combining their deterministic lookup with sender and receiver anonymity. Potential solutions include randomization of the lookup process [107, 109] and obfuscating the destination address [117]. These solutions are also of interest for hop-by-hop anonymization in F2F overlays.

We now shortly introduce methods to evaluate i) the impact of the forwarding decisions on the anonymity and ii) the confidentiality of the message content.

A method for i) consists of the identification of scenario for which the attacker can identify the sender or receiver. After identifying such scenarios, it is shown that they are in disagreement with the attacker's capabilities or their probability is low. In the latter case, the probability is expressed in relation to the fraction of controlled nodes. For instance, [163] uses the described methodology to characterize the anonymity of several anonymization schemes for DHTs.

A method for ii) assumes that encryption is used to hide information about the sender or receiver. Thus, the anonymity of the algorithm holds if the encryption is secure. Generally, the security of cryptographic algorithms is shown by reduction. More precisely, we use a proof of contradiction to show that if an attacker breaches the anonymity, it solves a (supposedly) difficult mathematical problem. Under the assumption that it is impossible or highly unlikely for the attacker to solve the problem, anonymity is achieved. [70] and [38] provide a formal introduction to reduction proofs and cryptography, respectively.

We focus on the evaluation of scalability, robustness, and censorship-resilience rather than on anonymity. Thus, we do not extend the existing methodology but mainly apply it. As stated in Section 2.6, we aim to design a local routing algorithms $\mathbf{R}_{node}$ for delivering a message from s to e without revealing the identity of s or e to any nodes on the path. The algorithm $\mathbf{R}_{node}$ relies upon the receiver information $info(e)$ in addition to the local state at each forwarding node. In order to guarantee possible innocence, we need to consider anonymity when designing the following components:

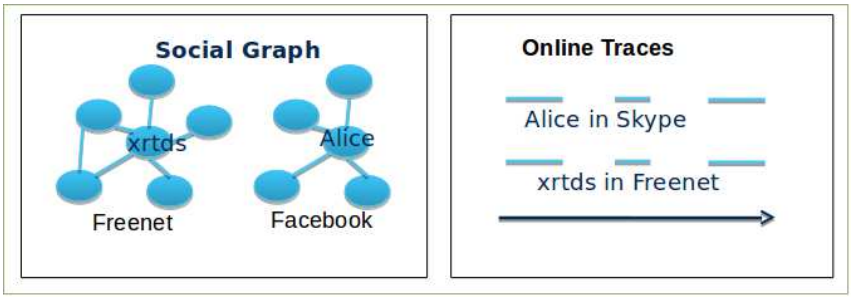


Figure 5.12: When revealing pseudonymous information in F2F overlays, external sources can be exploited to infer a node's identity and undermine the membership-concealment.

1. the local information stored at each node generated and maintained by **S**
2. the address generation algorithm $\mathbf{AdGen}_{node}$ creating $info(e)$, and
3. the routing algorithm $\mathbf{R}_{node}$.

In particular, we need to provide anonymity without losing the ability to discover routes efficiently.

5.5.2 Membership-Concealment

F2F overlays prevent direct communication between untrusted communication partners by relaying requests via a path of mutually trusted nodes. However, information gathered from forwarded requests and the behavior of neighbors can reveal the identity of participating users in the presence of external data sources such as the social graph or online traces. Figure 5.12 displays a simplified scenario allowing identification of users due to revealed pseudonymous data.

Incorporating all possible external data sources into our model to prove that the revealed information is insufficient to identify users is complex and out of scope for this thesis. We merely argue that we do reveal very little information about the structure of the social graph.

5.6 Discussion

We have introduced our methodology for the evaluation of F2F overlays. Two major concerns are associated with our chosen methodology, regarding on the one hand the efficiency and on the other hand the anonymization.

With regard to efficiency and scalability, our methodology is simplified as that it i) does not consider the routing efficiency during stabilization, and ii) does not consider the affect of concurrent topology changes. However, we address i) when evaluating the robustness by showing the effect of the fraction of failed nodes on the routing success. For ii), we developed a methodology for analyzing the likelihood if two topology changes affect one another. Based on the described methodology and realistic churn patterns, we ascertained that concurrent topology changes are unlikely and thus our methodology provides a valid approximation for our purposes.

In the context of the anonymization strategies, we concede the unsuitability of requiring only possible innocence for some scenarios. However, providing additional guarantees without either relying on additional limiting assumptions about the topology of the social graph or reducing the efficiency and robustness is challenging. For the initial approach, we thus only provide rather weak formal guarantees, leaving the design of additional security measures to future work. Note that our required degree of anonymity presents an improvement in contrast to deployed F2F overlays, which usually fail to provide any formal guarantees, as detailed in Chapter 3.

Chapter 6

Virtual Overlays

In this chapter, we evaluate the trade-off between routing and stabilization complexity in virtual overlays. In Section 3.3, we identified virtual overlays as the most promising state-of-the-art approach for F2F overlays. However, there is no evaluation about the long-time behavior of virtual overlays. In particular, it remains unclear if low stabilization complexity inherently leads to an increase in the routing complexity over an extended period of time.

Recall that virtual overlays aim to realize a structured overlay on top of a F2F overlay so that neighbors in the virtual overlay communicate by forwarding a request along a predefined path, called a tunnel or trail, in the F2F overlay. Thus, in the context of virtual overlays, stabilization is predominantly concerned with tunnel discovery and maintenance. Existing approaches either use flooding or leverage the routing protocol of the virtual overlay to discover a suitable tunnel endpoint. The new tunnel then corresponds to the discovered route to that endpoint, which potentially consists of multiple previously established tunnels, as recapitulated in Figure 6.1.

The state-of-the-art analysis shows that polylog routing and stabilization complexity can be achieved individually but does not provide a combined long-term evaluation of the two. As a consequence, we now consider the question if the two can be achieved simultaneously assuming that nodes frequently join and depart the overlay. Indeed, we show that virtual overlays *without an additional underlying routing scheme* are unable to provide both efficient stabilization and routing simultaneous.

In Section 6.1, we start by stating our results and proof ideas. We then present a formal model of virtual overlays and their evolution over time in Section 6.2. Afterwards, we first present our theoretical analysis in Section 6.3, followed by a brief simulation study in Section 6.4 with the goal of exemplary validating the asymptotic bounds. In Section 6.5, we conclude by stating the impact of our results on the remainder of this thesis. The presented results are published in INFOCOM 2015 [137].

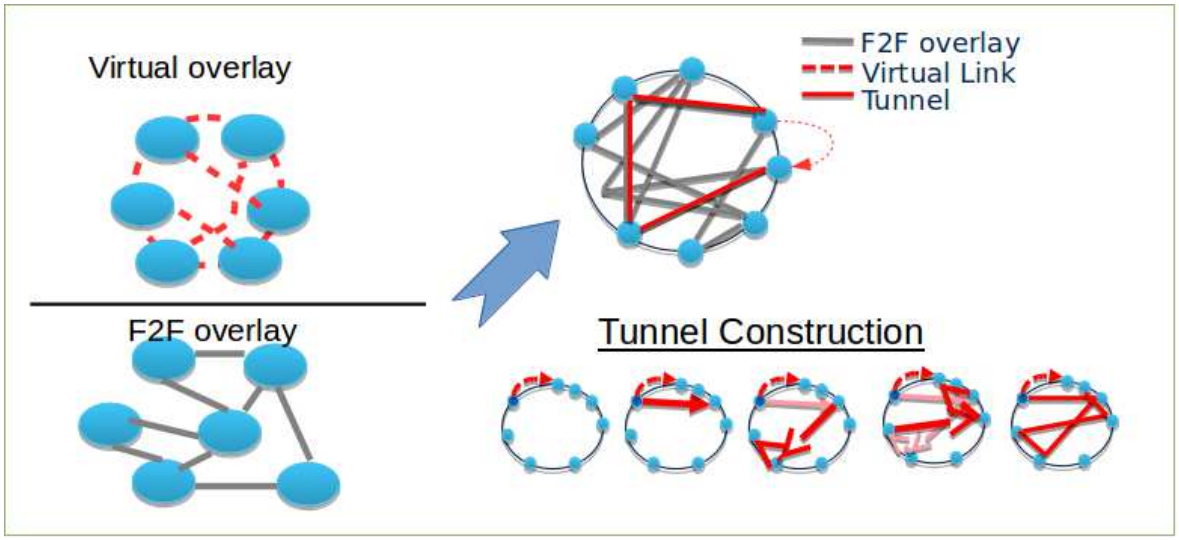


Figure 6.1: Recall the concept of virtual overlays from Section 3.3 (Figures 3.4 and 3.5): Virtual neighbors communicate via tunnels in the F2F overlay, tunnel construction usually concatenates existing tunnels

6.1 Statement of Result and Proof Idea

We first state our result and the proof idea in a very abstract fashion. Afterwards, we present some additional background and provide a more detailed explanation of the proof's steps. Note that this section is very informal and in parts oversimplifies to facilitate the understanding of the basic ideas.

In short, we show that virtual overlays cannot simultaneously provide polylog routing and stabilization complexity over an extended period of time. They thus fail to satisfy our requirements with regard to scalability.

Our theoretical analysis relies upon some assumptions, of which we now shortly motivate the most important ones. We assume that the F2F overlay offers no additional underlying routing protocol for communication between nodes. In other words, the tunnel construction and stabilization only uses local state information in form of i) the neighbor list of a node, and ii) records about the tunnels a node is contained in. As a consequence, we assume that a new tunnel indeed corresponds to a previously discovered route (or possibly one of several routes if the stabilization algorithm considers alternative routes). So, we assume that a new tunnel can only contain nodes that were also involved in the tunnel discovery. Using the actual route in the overlay for communication is different than the neighbor discovery in e.g., the Chord overlay. In Chord, the overlay routing is leveraged for neighbor discovery as well, but the actual connection is then established using IP routing [154]. In F2F overlays, IPs are only revealed to trusted contacts, so that IP routing between random overlay nodes is not an option. Nevertheless, similar addressing schemes could be realized in the F2F overlay in addition to virtual overlays. However, for the time being and in accordance with the existing approaches, we exclude alternative routing protocols.

In order to show that virtual overlays are inherently unable to provide polylog stabilization and routing complexity simultaneously, we assume that the expected stabilization complexity is polylog. Starting from some initial routing complexity, we consider the evolution of the routing complexity while nodes join and leave the overlay, resulting in the removals of old and the construction of new tunnels. Then, we prove that for all $r > 0$, the routing complexity eventually exceeds $\omega(\log^r n)$. Thus, the routing complexity does not remain polylog over an extended period of time if we restrict the stabilization complexity. The idea of the proof is to show that newly established tunnels are in expectation longer than tunnels that are removed. In the light of the above assumptions, we see that new tunnels generally consist of (parts of) old tunnels. Intuitively, concatenating several existing tunnels results in longer than average tunnels, as illustrated in Figure 6.2, and thus the expected tunnel length should increase over time. As routing concatenates $\mathcal{O}(\log n)$ tunnels, the routing exceeds polylog complexity if the tunnels exceed polylog length in expectation. In the proof, we formalize the above intuition, i.e., we show that in expectation the tunnel length and hence the routing complexity increases over time if the expected stabilization complexity is polylog. Thus, if we utilize virtual overlays without an additional routing scheme, we have to either allow a higher stabilization complexity or cannot achieve the required routing complexity.

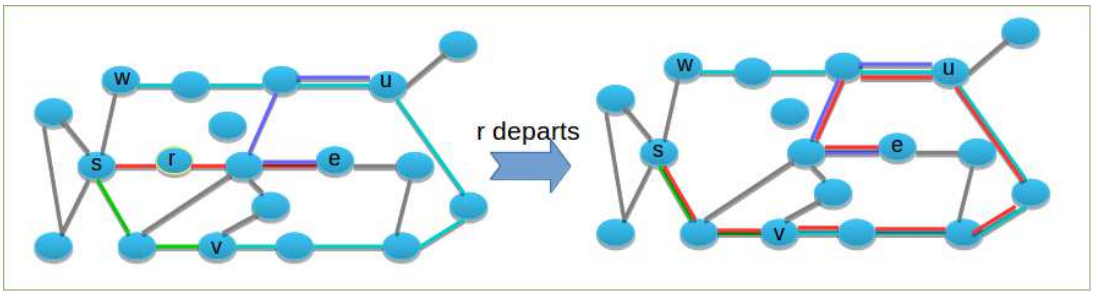


Figure 6.2: Illustrating the proof idea that the tunnel length increases over time: Tunnels, indicated here by colored rather than gray lines, are replaced by new tunnels concatenating (parts of) existing tunnels, thus in expectation increasing the average tunnel length. Here, the new tunnel from s to e (red) consists of the tunnel from s to v (green) as well as parts of the tunnels from v to w (cyan) and u to e (blue)

We model the evolution of a virtual overlay, and in particular the distribution of the tunnel length, as a discrete stochastic process. For this purpose, we follow the methodology described in Section 5.2.3 with two slight modifications. First, we consider the tunnel length rather than the actual routing complexity. As argued above, the tunnel length is a lower bound on the routing complexity. Hence, it suffices to show that the tunnel length does not scale polylog in the number of nodes. Second, when nodes join or depart the overlay, several tunnels are affected. For instance, a new node u needs to establish multiple tunnels and old nodes might have to replace some of their tunnels with tunnels to u. Similarly, if a node v departs, all tunnels containing v either entirely cease to exist or are replaced by alternative tunnels. The number of affected tunnels per topology change is hard to determine, especially since more reliable

nodes are likely to be contained in more than the average number of tunnels. So, the number of affected tunnels does not correspond to the average number of tunnels per node. As a consequence, we model a step of the stochastic process as a *tunnel change*, i.e., the removal or construction of a tunnel, rather than a *topology change*. Because each topology change results in at least one tunnel change, the number of tunnel changes represents a lower bound on the number of topology changes.

The above model motivates a straight-forward proof strategy: derive an upper bound l_t on the length of a removed tunnel at step t, i.e., after t tunnel changes, and a lower bound u_t on the length of a newly constructed tunnel such that $l_t < u_t$. Then the tunnel length is guaranteed to increase over time. However, the actual proof is slightly more complex for two reasons:

1. The probability of a tunnel to be affected by a topology change increases with the length of the tunnel. If the tunnel is removed due to a node departure, the probability of a tunnel to contain the departed node increases linearly with the tunnel length. In addition, overlays like X-Vine strategically replace long tunnels. Thus, the expected length of a removed tunnel is potentially longer than the average tunnel length. In order to give bounds on the length of removed tunnels, we hence need some information about the tunnel length distribution rather than only the expected tunnel length.

2. In order to show that the inability to combine efficient stabilization and routing is an inherent trait of virtual overlays, we need to show the claim for arbitrary virtual overlay topologies and stabilization algorithms. Such a high degree of generality complicates the derivation of bounds on the length of new tunnels. The only assumption about the nature of the new tunnel is that only nodes previously involved in the tunnel discovery are included in the tunnel, which allows a broad range of both probabilistic and deterministic algorithms. Similarly, our only assumption about the topology of the virtual overlay is that routing requires the traversal of $\mathcal{O}(\log n)$ tunnels, which again leaves a high degree of freedom with regard to the (virtual) neighbor selection.

Thus, despite the simplicity of the proof idea, the details of the proof require careful consideration due to the generality of the model.

One of our key difficulties described above is the generalization to arbitrary overlay structures instead of considering specific realizations such as rings or trees. Rather than showing the result for all overlays in general, we differentiate two classes of overlays and show the claim for each class individually. First, we consider fully determined overlays. Informally, a fully determined overlay is characterized by uniquely defined tunnel endpoints. For instance, each node in a ring has a connection to its uniquely defined predecessor and successor. Hence, given the set of coordinates, the topology of the virtual overlay is uniquely defined. Second, we consider partially determined overlays, which restrict the set of potential endpoints to a certain set of nodes but not necessarily to one node. For instance, Kademlia restricts the set of endpoints of a tunnel to nodes whose coordinates have a certain common prefix length. Due to their diversity with regard to the (virtual) neighbor selection, fully and partially determined overlays require different proof strategies.

We now shortly describe the steps of the proof for fully determined virtual overlays. Note that the expected value of a non-negative random variable X with q-quantile x, i.e., the lowest value x such that $P(X \leq x) \geq q$, is bound by $\mathbb{E}(X) \geq (1-q) \cdot x$. We first show that for some q, the q-quantile of the tunnel length distribution increases by a least 1 every k steps. For this purpose, we relate the probability of a removed tunnel to be shorter than the q-quantile x to the probability of a new tunnel to be shorter than x. For the latter, we determine the probability that a new tunnel is not a concatenation of tunnels of length less than x. If the q-quantile of the tunnel length distribution increases by 1 each k steps, we have a q-quantile of at least $\log^r n$ after $k \cdot \log^r n$ steps and thus an expected tunnel length of at least $(1-q)\log^r n$. In this manner, we do not only show that the expected tunnel length exceeds $\omega(\log^r n)$ eventually but also derive an upper bound on the expected number of tunnel changes until the polylog bound does no longer hold.

The proof idea for partially determined overlays is slightly more complex. We first divide the tunnels into classes $F_{t,1}, \ldots, F_{t,m}$ according to the distance between their start- and endpoint's coordinate. Afterwards, we argue that the fraction of tunnels within each class has to remain roughly constant for efficient routing. If the fractions indeed remain roughly constant for an extended period of time, we can then express the expected tunnel length within class $F_{t,i}$ recursively in terms of the expected tunnel length of the classes $F_{t,j}$ for $j > i$ and $t \to \infty$. In order to derive the recursive formula, we prove that a newly constructed tunnel in $F_{t,i}$ contains one tunnel from each $F_{t,i+1}, \ldots, F_{t,m}$ with high probability. In the last step, we resolve the recursive formula to show that the expected routing length for $F_{t,0}$ is eventually not polylog, i.e., exceeds $\omega(\log^{r+1} n)$ for all $r > 0$. Because $F_{t,0}$ contains at least a fraction $\Omega(1/\log n)$ of all tunnels, it follows that the expected tunnel length is $\omega(\log^r n)$. Note that we do not determine the number of tunnel changes until the polylog bound does no longer hold but only show that at some point

in time, the routing exceeds polylog complexity. Nevertheless, we show that efficient stabilization and routing in partially determined overlays are mutually exclusive.

We have now described our theoretical evaluation of virtual overlays in a nutshell. The main idea lies in showing that the concatenation of tunnels results in an increased tunnel length. If nodes join and depart the overlay over an extended period of time, the tunnel length continues to increase until the routing complexity exceeds the desired polylog bound. A rigorous model and subsequent proof follow.

6.2 Modeling Virtual Overlays as a Stochastic Process

In this section, we formally model the tunnel length over time. We start by formalizing the concept of virtual overlays, especially their evolution over time. Afterwards, we recapitulate the concept of a metric from Section 5.1.2, considering in particular multidimensional metrics. Last, we state our assumptions for the subsequent theoretical analysis.

Recall that all our algorithms are distributed and local, meaning that each node bases its actions purely on its partial view of the network.

6.2.1 Notation

We start by defining a static virtual overlay as a tuple, then we consider a dynamic virtual overlay.

(Static) Virtual Overlay: A *virtual overlay* is a 7-tuple $O = (V, E, \mathbf{X}, id, F, \mathbf{R}, \mathbf{S})$, so that

- (V, E) is a connected graph with node set V and edge set $E \subset V \times V$.
- $\mathbf{X}$ is a finite (semi-)metric space with a distance function δ_X.
- $id\colon V \to \mathbf{X}$ is the coordinate assignment. Coordinates are chosen independently of the underlying graph (V, E), so that there is no relation between the distance of two nodes' coordinates and the length of the (shortest) paths between them.
- $F \subset V^*$ is the tunnel set, consisting of vectors $p = (v_0, v_1, \dots, v_l)$ for some $l \in \mathbb{Z}_n$ with $(v_i, v_{i+1}) \in E$ for $i = 0, \dots, l-1$.
- $\mathbf{R}$ is a local *routing algorithm* that, given an arbitrary coordinate $w \in \mathbf{X}$ and source node $s \in V$, finds a path from s to $e \in V$ such that $\delta_X(w, id(e))$ is minimized over all nodes.
- $\mathbf{S}$ denotes the stabilization algorithm. During the analysis, we are only concerned with its subroutine $\mathbf{T}$, a local distributed *tunnel discovery algorithm*. Given a source node v_0, $\mathbf{T}$ finds a tunnel $p = (v_1, \dots, v_l)$ to a virtual overlay neighbor v_l.

We adapt the following notation throughout this chapter. Note that the distance function δ_X is defined for elements of $\mathbf{X}$. For a simplified notation, we extend the definition to nodes $v, u \in V$, i.e., $\delta_X(v, u) = \delta_X(id(u), id(v))$. In the following, we refer to the first and the last hop of a tunnel p as *startpoint* $s(p)$ and *endpoint* $e(p)$, respectively. We define the length $|p|$ of a tunnel p as the number of *nodes*[1] on the tunnel and assume tunnels to be acyclic. A node v is said to be *contained* in a tunnel $p = (v_0, v_1, \dots, v_l)$ if $v = v_i$ for some $i = 0 \dots l$. Last, the tunnel length distribution L, our main metric of interest, gives the fractions of tunnels of length i for all $i \in \mathbb{Z}_n$.

State Information: Nodes maintain local state information in order to process requests. Each node $v \in V$ in a virtual overlay O keeps a *neighbor set* $NT(v)$

$$NT(v) = \{w \in V, (v, w) \in E\},$$

a *routing table* $RT(v)$ of tunnels with startpoint v

$$RT(v) = \{(id_e, v_2) \in \mathbf{X} \times V\colon \exists f = (v, v_2, \dots, v_l) \in F, id(v_l) = id_e\}$$

and a *tunnel table* $FT(v)$ of tunnels v is contained in but not the start- or endpoint

$$\begin{aligned} FT(v) = \{&(id_s, id_e, v_-, v_+) \in \mathbf{X} \times \mathbf{X} \times V \times V\colon \\ &\exists f = (v_1, \dots, v_-, v, v_+, \dots, v_l) \in F, id(v_0) = id_s, id(v_l) = id_e\}. \end{aligned}$$

Figure 6.3 illustrates the three types of state information.

[1] In contrast to common path length, which gives the number of edges, as using the number of nodes simplifies some of the following equations

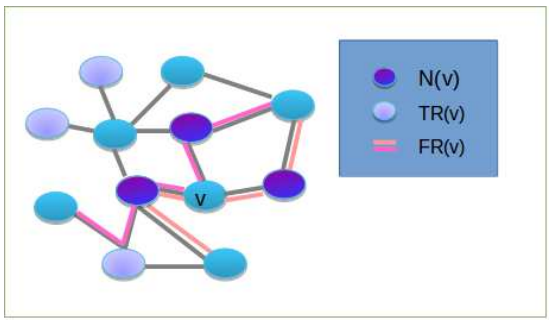


Figure 6.3: State information in virtual overlays: node v maintains list of F2F overlay neighbors $N(v)$, virtual overlay neighbors $RT(v)$, and tunnels $FT(v)$ that v is contained in

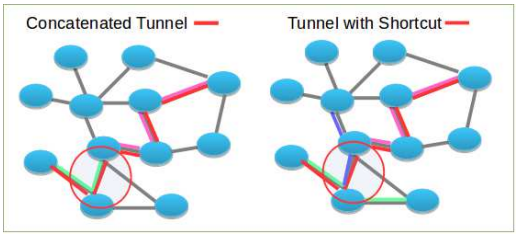


Figure 6.4: New Tunnels can be pure concatenations of tunnels (left: red tunnel is concatenation of light pink and green tunnel) or contain shortcuts, i.e., only contain parts of certain tunnels (right: red tunnel contains light pink tunnel but only parts of the blue and green tunnels)

Routing and Tunnel Discovery: For both the routing algorithm $\mathbf{R}$ as well as the tunnel discovery algorithm $\mathbf{T}$, a node v contacts a set $next \subset NT(v)$ of its neighbors based on $RT(v)$ and $FT(v)$. If all nodes of an old tunnel $p \in F$ are contained in a newly constructed tunnel p', we say that p is *contained* in p'. We say that a newly constructed tunnel $p' = (v_0, \ldots, v_l)$ contains a *shortcut* if p' cannot be represented as a concatenation of old tunnels. We give examples for concatenated tunnels and tunnels with shortcuts in Figure 6.4.

Fully and Partially Determined Overlays: As mentioned in Section 6.1, we differentiate two types of virtual overlays, fully determined and partially determined overlays. We illustrate the difference between the two overlay types in Figure 6.5, in agreement with Definition 6.1.

Definition 6.1. *A virtual overlay $O = (V, E, \mathbf{X}, id, F, \mathbf{R}, \mathbf{S})$ is called fully determined if the set tunnel set F is uniquely determined by $(V, E, \mathbf{X}, id)$. Otherwise, O is partially determined.*

The above definition of partially determined overlays is very general. We limit our analysis to approaches that allow for routing using polylog virtual links. We present a more exact definition of such overlays in Section 6.3.3, which is motivated by Kleinberg's result on the routing complexity in multi-dimensional lattices [86]. For the time being, we focus on general concepts, predominantly the evolution of virtual overlays over time.

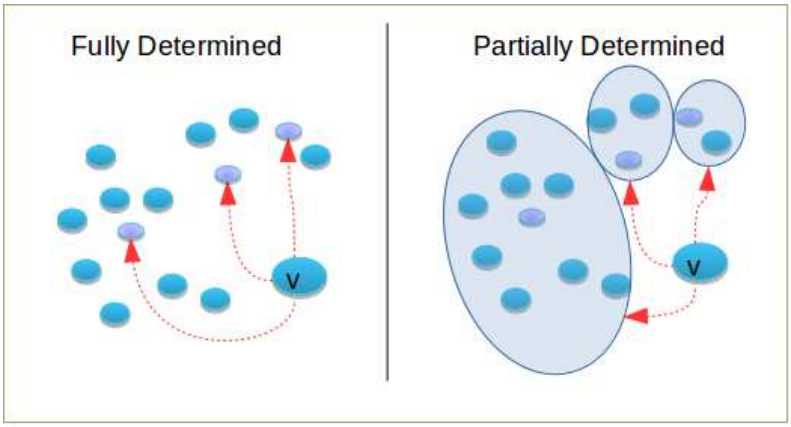


Figure 6.5: Fully determined and partially determined overlays: The endpoint of a tunnel can either be uniquely determined by the set of nodes and the coordinate assignment id or not. If not, the set of potential endpoints is usually a real subset of all nodes determined by the coordinate assignment id.

Dynamic Virtual Overlay: Now, we model the evolution of a virtual overlay or, more precisely, of the tunnel length in a virtual overlay, as a discrete random process. Each step of the random process corresponds to either establishing or removing one tunnel. Since we assume the network size to remain largely constant, both are equally likely. Thus, we slightly deviate from the methodology discussed in Section 5.2.3 for two reasons. As motivated in Section 6.1, we consider the tunnel length rather than the actual routing length. Furthermore, a step of the stochastic process corresponds to the removal or construction of a tunnel rather than a topology change. However, the tunnel length and the number of

tunnel changes present lower bounds for the routing length and the number of topology changes, so that any lower bounds on the former two are also lower bounds on the the latter two.

The state of the virtual overlay at time t is denoted by $O_t = (V_t, E_t, \mathbf{X}, id, F_t, \mathbf{R}, \mathbf{S})$. Analogously, the neighbor set, routing table, and tunnel table at step t are called $NT_t(v)$, $RT_t(v)$, and $FT_t(v)$, respectively. The evolution of the tunnel length distribution is modeled as a random process $(L_t^{\mathbf{T}})_{t\in\mathbb{N}}$, with $L_t^{\mathbf{T}}$ being the tunnel length distribution after t tunnels have been changed. We denote by $NL_t^{\mathbf{T}}$ the length of a newly constructed tunnel at step t, and by $RL_t^{\mathbf{T}}$ the length of a removed tunnel.

6.2.2 Multidimensional Metrics

Note that realizations of the random variable $L_t^{\mathbf{R}}$ are discrete distributions rather than single real values. So, each realization of $L_t^{\mathbf{R}}$ represents a random variable and thus a probability mass function. Hence, each realization of the random process $(L_t)_{t\in\mathbb{N}_0}$ is a sequence of probability mass functions. In order to express characteristics of $(L_t)_{t\in\mathbb{N}_0}$, we want to apply real-valued functions such as the computation of its mean at certain points in time. So, to avoid confusions between the expected value over time and the mean at a certain point in time, we distinguish between the two in notation.

We apply functions to $(L_t)_{t\in\mathbb{N}_0}$ as follows. As we assume tunnels to be acyclic, the tunnel length is bound by $n-1$. Hence, $L_t^{\mathbf{R}}$ attains probability mass functions on $\mathbb{Z}_n = \{0,\dots,n-1\}$. Denote by $\Pi(\mathbb{Z}_n)$ the set of all probability mass functions on $\mathbb{Z}_n$. For any $x \in \Pi(\mathbb{Z}_n)$, we denote the probability that x has value i by $x(i)$. The *mean* of x is denoted by $mean(x) = \sum_{i=0}^{n-1} ix(i)$. Consider that a realization of $L_t^{\mathbf{R}}(i)$ gives the fraction of tunnel of length i. Because the number of such pairs is $n(n-1)$, all probabilities are a multiple of $\frac{1}{n(n-1)}$, so that $L_t^{\mathbf{R}}(i)$ only takes a finite number of values in $\mathbb{Z}_n = \{0,\dots,n-1\}$. This observation ensures that our definition of the expectation is well-defined: the *expectation* at time t of any function $f\colon \Pi(\mathbb{Z}_n) \to \mathbb{R}$ on the random variable $L_t^{\mathbf{R}}$ is

$$\mathbb{E}(f(L_t^{\mathbf{R}})) = \sum_{x\in\Pi(\mathbb{Z}_n)} f(x)P(L_t^{\mathbf{R}} = x).$$

In particular, the expected mean is given by

$$\mathbb{E}(mean(L_t^{\mathbf{R}})) = \sum_{x\in\Pi(\mathbb{Z}_n)} mean(x)P(L_t^{\mathbf{R}} = x).$$

Furthermore, for all function $g\colon \mathbb{R} \to \mathbb{R}$ and $x \in \Pi(\mathbb{Z}_n)$, we define

$$mean(g(x)) = \sum_{i=0}^{n-1} g(i)x(i), \tag{6.1}$$

and thus

$$\mathbb{E}(mean(g(L_t^{\mathbf{R}})) = \sum_{x\in\Pi(\mathbb{Z}_n)} P(L_t^{\mathbf{R}} = x) \sum_{i=0}^{n-1} g(i)x(i). \tag{6.2}$$

So, we can use the above terminology to model the evolution of multidimensional graph properties, in particular the tunnel length.

6.2.3 Assumptions

We set $n = |V_0|$ and assume that the overlay is stable, i.e., there exists no drift towards an increased or decreased overlay size and the expected number of nodes at any point in time is n. In addition, we make the following assumptions:

1. The average degree $2\frac{|E_t|}{|V_t|} \leq K$ is bound by a constant K.

2. $|F_t| = \theta\left(n \log^{\alpha} n\right)$ for some $\alpha \in \mathbb{R}$, i.e. the average routing table size is polylog.

3. During the execution of the tunnel discovery algorithm $\mathbf{T}$ at most $\mathbb{E}(CS_t^{\mathbf{S}}) = \mathcal{O}\left(\log^{\beta} n\right)$ messages are exchanged for stabilization, including in particular the messages required by the tunnel discovery algorithm $\mathbf{T}$.

Assumption 1 of a constant average degree can be replaced by the assumption of a polylog average degree. However, such an increased bound requires the use of an additional variable and thus further complicates the notation. For simplicity, we thus choose the above assumption, which is common for social networks, which correspond to the F2F overlay topology. Assumption 2 holds for $\alpha = 1$ for most common structured overlays such as Chord and Kademlia. Anyways, as the number of tunnels per node is a lower bound on the stabilization overhead per node join, polylog overhead cannot be achieved if Assumption 2 does not hold. Assumption 3 states that we only consider algorithms with polylog stabilization complexity to show that these are not sufficient for maintaining short tunnels, i.e., we apply a proof by contraposition.

6.3 Theoretical Analysis

We now prove the claim that efficient stabilization of the tunnels is not possible over an extended period of time. For this purpose, we first give some general results about tunnel discovery. Afterwards, we consider fully determined and partially determined overlays individually.

6.3.1 Basic Properties

In this section, we i) define the distribution of $RL_t^{\mathbf{T}}$, the length of a removed tunnel, and ii) obtain an upper bound on the probability of a tunnel to contain a shortcut.

We start by expressing the distribution of $RL_t^{\mathbf{T}}$, the length of a removed tunnel, in terms of current tunnel length distribution $L_t^{\mathbf{T}}$. There are two reasons to remove a tunnel: either a node on the tunnel leaves the overlay or an alternative shorter tunnel was found. Denote the first event by A_1 and the second by A_2. The overall stabilization complexity per topology change is $\mathcal{O}(\log^\beta n)$. As each removed node results in at least one removed tunnel, the probability of event A_1 is at least

$$P(A_1) = \Omega\left(\frac{1}{\log^\beta n}\right). \tag{6.3}$$

Now, we consider the probability for removing a tunnel of a certain length if A_1 holds. Let $l_t^{\mathbf{T}}$ be a realization of the corresponding tunnel length distribution. For any tunnel p of length $|p|$, denote by $x(p)$ the event that the tunnel p is affected by a node removal. Furthermore, let $|p|$ is the length of p. A tunnel of length i is destroyed if any of its i nodes leave, i.e.,

$$P(x(p)||p| = i \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1) = \frac{i}{n}.$$

The probability that a removed tunnel p has length i given the realization $l_t^{\mathbf{T}}$ is

$$\begin{aligned}
&P\left(|p| = i | x(p) \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1\right) = \\
&\frac{P\left(x(p)||p| = i \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1\right) P\left(|p| = i | L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1\right)}{P\left(x(p)||p| = i \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1\right)} \\
&= \frac{i \cdot l_t^{\mathbf{T}}(i)}{n \sum_{j=0}^{n} \frac{j \cdot l_t^{\mathbf{T}}(j)}{n}} = \frac{i \cdot l_t^{\mathbf{T}}(i)}{mean(l_t^{\mathbf{T}})}.
\end{aligned}$$

applying Bayes' rule in the first step. We thus define the probability that a removed tunnel at time t given A_1 has length i as the expectation of $P\left(|p| = i | x(p) \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1\right)$ over all possible realizations $l_t^{\mathbf{T}}$ of $L_t^{\mathbf{T}}$

$$\begin{aligned}
&P(\, RL_t^{\mathbf{T}} = i \big|\, A_1) = \mathbb{E}\left(\frac{iL_t^{\mathbf{T}}(i)}{mean(L_t^{\mathbf{T}})}\middle|\, A_1\right) \\
&= \sum_{l_t^{\mathbf{T}} \in \Pi(\mathbb{Z}_n)} P(|p| = i | x(p) \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1) P(L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1).
\end{aligned} \tag{6.4}$$

By Equation 6.3, the overall probability that a removed tunnel has length i is at least

$$\begin{aligned}
&P(RL_t^{\mathbf{T}} = i) = \Omega\left(\frac{1}{\log^\beta n} \mathbb{E}\left(\frac{iL_t^{\mathbf{T}}(i)}{mean(L_t^{\mathbf{T}})}\middle|\, A_1\right)\right) \\
&= \Omega\left(\frac{1}{\log^\beta n} \sum_{l_t^{\mathbf{T}} \in \Pi(\mathbb{Z}_n)} P(|p| = i | x(p) \cap L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1) P(L_t^{\mathbf{T}} = l_t^{\mathbf{T}} \cap A_1)\right).
\end{aligned} \tag{6.5}$$

Now, we show that the probability to shortcut to the destination such that none of the existing tunnels is fully contained in a newly constructed tunnel is small. More precisely, an endpoint $e(p')$ of a newly constructed tunnel p' is either found when a tunnel leading to $e(p')$ is fully contained in p' or if p' contains a shortcut to $e(p')$. We obtain an upper bound on the probability of the latter. In the following sections, we then show that a new tunnel is a concatenation of old tunnels with high probability and hence likely to be longer than existing tunnels.

Lemma 6.2. *Let $O_t = (V_t, E_t, \mathbf{X}, id, F_t, \mathbf{R}, \mathbf{S})$ be a virtual overlay with tunnel discovery algorithm $\mathbf{T}$. Set $n = |V_0|$ and assume $|F_t| = \mathcal{O}(n \log^\alpha n)$. Furthermore, let M_t be the number of messages exchanged during a tunnel discovery. The probability of the event H that the newly constructed tunnel contains a shortcut to at least one of Z_t nodes is bound from above by*

$$P(H) = \mathcal{O}\left(\frac{\mathbb{E}(M_t)\left(\mathbb{E}(mean(L_t^{\mathbf{T}}))\log^\alpha n + 2\mathbb{E}\left(\frac{E_t}{V_t}\right)\right)\mathbb{E}(Z_t)}{n}\right). \tag{6.6}$$

Proof. We first show a version of Jensen's inequality for random variables in $\Pi(\mathbb{Z}_n)$. In the main part of the proof, we then obtain the probability that one node does not have a shortcut to any of the Z_t nodes. The probability that M_t nodes cannot provide a shortcut follows from a union bound.

We modify Jensen's inequality, which states that for a convex real-valued function $f : \mathbb{R} \to \mathbb{R}$, i.e., a function with $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$, and an integrable real-valued random variable X,

$$f(\mathbb{E}(X)) \leq \mathbb{E}(f(X)). \tag{6.7}$$

In particular, a twice differentiable function f with non-negative second derivative is convex. A proof of Equation 6.7 can be found in [90]. We now show the following version for discrete random variables X with finitely many values in $\Pi(\mathbb{Z}_n)$: For any convex function $g\colon \mathbb{R} \to \mathbb{R}$, it holds that

$$\mathbb{E}(mean(g(X))) \geq g(\mathbb{E}(mean(X)). \tag{6.8}$$

For the proof, let $x \in \Pi(\mathbb{Z}_n)$. By the definition of mean and Equation 6.7, we have $mean(g(x)) \geq g(mean(x))$. Furthermore, by Equation 6.2

$$\mathbb{E}(mean(g(X))) = \sum_{x\in\Pi(\mathbb{Z}_n)} P(X = x)mean(g(x)) \geq \sum_{x\in\Pi(\mathbb{Z}_n)} P(X = x)g(mean(x)).$$

Consider a real-valued random variable $\tilde{X}$ with $P(\tilde{X} = mean(x)) = P(X = x)$ for all $x \in \Pi(\mathbb{Z}_n)$. Then

$$\begin{aligned}&\sum_{x\in\Pi(\mathbb{Z}_n)} P\left(\tilde{X} = mean(x)\right) g(mean(x))\\ =&\mathbb{E}\left(g(\tilde{X})\right) \geq g\left(\mathbb{E}(\tilde{X})\right) = g\left(\mathbb{E}(mean(X))\right).\end{aligned}$$

The second last step follows from Equation 6.7, the last step from the definitions of $\tilde{X}$ and $\mathbb{E}(mean(X))$. This completes the proof of Equation 6.8.

Now, we prove the claim in Equation 6.6. A node v is aware of its neighbors in the underlay, as well as the startpoints and endpoints of the tunnels it is contained in. Denote by H_1 the event that a node v does not have a shortcut to any of the Z_t potential endpoints. If Z_t takes value z and v has d neighbors and is contained in y tunnels, the probability that v is not aware of any possible endpoint is at most $(1 - \frac{z}{n})^{d+y}$. The mean number of tunnel table entries Y_t per node is at most

$$\mathbb{E}(Y_t) = \mathcal{O}\left(\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right)\log^\alpha n\right)$$

because there are $\mathbb{E}(|F_t|) = \mathcal{O}(n \log^\alpha n)$ tunnels of expected average length $\mathbb{E}(mean(L_t^{\mathbf{T}}))$.

We apply Jensen's inequality and Equation 6.8 for the convex functions $f_1(z) = 1 - \frac{z}{n}$ and $f_2(x) = \left(1 - \frac{\mathbb{E}(Z_t)}{n}\right)^x$ to obtain

$$P(H_1) \geq \mathbb{E}\left(\left(1 - \frac{Z_t}{n}\right)^{|N_t(v)|+Y_t}\right) \geq \left(1 - \frac{\mathbb{E}(Z_t)}{n}\right)^{2\mathbb{E}\left(\frac{E_t}{V_t}\right)+\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right)\log^\alpha n}.$$

Hence, the probability $P(H)$ can be bound as the complement of the event that none of at most M_t nodes contacted during the tunnel discovery are aware of a potential endpoint. Again, we apply Jensen's inequality to the function

$$g(x) = \left(\left(1 - \frac{\mathbb{E}(Z_t)}{n}\right)^{2\mathbb{E}\left(\frac{E_t}{V_t}\right)+\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right)\log^\alpha n}\right)^x,$$

resulting in

$$P(H) \leq 1 - \mathbb{E}(g(M_t)) \leq 1 - g(\mathbb{E}(M_t)) =$$
$$1 - \left(1 - \frac{\mathbb{E}(Z_t)}{n}\right)^{\left(2\mathbb{E}\left(\frac{E_t}{V_t}\right)+\mathbb{E}(mean(L_t^{\mathbf{T}}))\log^{\alpha} n\right)\mathbb{E}(M_t)} =$$
$$\mathcal{O}\left(\frac{\mathbb{E}(M_t)\left(\mathbb{E}(mean(L_t^{\mathbf{T}})\log^{\alpha} n + 2\mathbb{E}\left(\frac{E_t}{V_t}\right)\right)\mathbb{E}(Z_t)}{n}\right)$$

as claimed. □

We now prove the main result of this chapter, the impossibility to efficiently self-stabilize virtual overlays with inherent dynamics. We start with the slightly simpler scenario of fully determined overlays, before considering less restricted partially determined overlays.

6.3.2 Fully Determined Overlays

In this section, we consider *fully determined* virtual overlays, for which the tunnel start- and endpoints are uniquely determined by the coordinate assignment. For example, a virtual overlay based on Chord is fully determined. We show that for all $r > 0$, the expected mean tunnel length is at least of order $\log^r n$ after $n \log^{3r+\alpha+2\beta+1} n$ steps of the random process described in Section 6.2.

Theorem 6.3. *Let $O_t = (V_t, E_t, \mathbf{X}, id, F_t, \mathbf{R}, \mathbf{S})$ be a fully determined virtual overlay with tunnel discovery algorithm $\mathbf{T}$. For any $r > 0$, the expected mean tunnel length is bound from below by*

$$\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \Omega(\log^r n) \text{ for all } t = \Omega(n \log^{3r+\alpha+2\beta+1} n).$$

Proof. The idea of the proof is to show that the q-quantile of $L_t^{\mathbf{T}}$ for a suitable q increases beyond polylog. Thus, we determine the probability for added and removed tunnels each to be shorter than the quantile. The increase in the q-quantile then arises from the fact that removing a tunnel of such a length is more likely than adding a new tunnel. Hence, the average tunnel length increases beyond polylog eventually. The bound on the actual number of tunnel changes until the tunnel length exceeds a certain polylog bound is obtained by comparing the probabilities for adding and removing such tunnels. From those probabilities, we derive an upper bound on the expected number x of steps until the q-quantile is increased by 1. If the expected value of the q-quantile increases by 1 in x steps independently of its actual value, it increases by $\log^r n$ in $x \log^r n$ steps.

Let $\lambda_t(q)$ be the q-quantile of $L_t^{\mathbf{T}}$ for some $q = \frac{1}{\log^k n}$ where $k > 1$ is determined during the proof. In the following, we bound the number of steps until $\mathbb{E}(\lambda_t(q)) = \Omega(\log^r n)$. Then the expected mean tunnel length is at least of order $\log^r n$ as well because for $n \geq 4$,

$$\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) \geq (1-q)\mathbb{E}(\lambda_t(q)) > \frac{\mathbb{E}(\lambda_t(q))}{2} = \Omega(\log^r n).$$

Let C_t be the number of tunnels that are at least of length $\lambda_t(q)+1$. We show that the expected increase in C_t is

$$\mathbb{E}(C_t - C_{t-1}) = \Omega\left(\frac{q}{\log^{r+\beta} n}\right), \tag{6.9}$$

independent of t. Based on Equation 6.9, we can determine the number of steps needed to increase the q-quantile by 1. If indeed Equation 6.9 holds, the expected number of tunnels with length at least $\lambda_t(q)+1$ increases by $\Omega\left(\frac{q}{\log^{r+\beta} n}\right)$ for one tunnel addition or removal. As a consequence, after x changes, the number of such tunnels increases by $\Omega\left(x\frac{q}{\log^{r+\beta} n}\right)$. Hence, there are $(1-q)|F_t|$ tunnels of length at least $\lambda_t(q)+1$ after $\mathcal{O}\left((1-q)\frac{\log^{r+\beta} n}{q}|F_t|\right) = \mathcal{O}\left(\frac{\log^{r+\beta} n}{q}|F_t|\right)$ changes, i.e.,

$$\forall t_0 \colon \mathbb{E}(\lambda_{t_0+t}(q)) = \Omega\left(\mathbb{E}(\lambda_{t_0}(q)) + 1\right) \text{ for all } t = \Omega\left(\frac{\log^{r+\beta} n}{q}|F_{t_0}|\right).$$

An upper bound on the number of steps to increase the mean tunnel length by $\log^r n$ follows directly. It is

$$\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \Omega\left(\mathbb{E}(\lambda_t(q))\right) = \Omega\left(\mathbb{E}(\lambda_0(q)) + \log^r n\right) = \Omega\left(\log^r n\right) \tag{6.10}$$
$$\text{for all } t = \Omega\left(\log^r n \frac{\log^{r+\beta} n}{q}|F_0|\right) = \Omega\left(n\frac{\log^{2r+\alpha+\beta} n}{q}\right).$$

It remains to prove Equation 6.9. If a new tunnel of length at least $\lambda_t(q)+1$ is constructed, the number of such tunnels increases by 1, and decreases by 1 if such a tunnel is removed. Removal and construction are equally likely, so that $\mathbb{E}(C_t - C_{t-1}) = \frac{1}{2}\left(P\left(NL_t^{\mathbf{T}} > \lambda_t(q)\right) - P\left(RL_t^{\mathbf{T}} > \lambda_t(q)\right)\right)$.

We assume $\mathbb{E}(mean(L_t^{\mathbf{T}})) = \mathcal{O}(\log^r n)$, otherwise the claim holds. Each removed tunnel is of length at least 1, so that by Equation 6.5

$$P(RL_t^{\mathbf{T}} \leq \lambda_t(q)) = \Omega\left(\frac{q}{\log^\beta n \mathbb{E}(mean(L_t^{\mathbf{T}}))}\right) = \Omega\left(\frac{q}{\log^{r+\beta} n}\right). \tag{6.11}$$

In order to bound the tunnel length of a newly constructed tunnel, consider the event H that the discovered tunnel p contains less than two old tunnels. Otherwise, the new tunnel can only be of length at most $\lambda_t(q)$ if all contained tunnels are shorter than $\lambda_t(q)$. The probability that all tunnels are shorter than $\lambda_t(q)$ decreases exponentially with the number of contained tunnels, so that the case of two tunnels presents an upper bound. Hence, we get

$$\begin{aligned} P(NL_t^{\mathbf{T}} \leq \lambda_t(q)) &= P(NL_t^{\mathbf{T}} \leq \lambda_t(q)|H)P(H) + P(NL_t^{\mathbf{T}} \leq \lambda_t(q)|H^\perp)(1-P(H)) \\ &\leq P(H) + P(NL_t^{\mathbf{T}} \leq \lambda_t(q)|H^\perp) \leq P(H) + q^2. \end{aligned} \tag{6.12}$$

The probability $P(H)$ in Equation 6.12 can be bound by Lemma 6.2. We determine an upper bound on the number of nodes Z_t, so that p can only contain less than two tunnels if it contains a shortcut to any of those Z_t nodes. Let u be the uniquely determined endpoint of the new tunnel. Then the Z_t nodes consist of all nodes on a tunnel to u or to any of u's overlay neighbors.

We hence define

$$ON_t(u) = \{v \in V \colon \exists (id(u), v_2) \in RT(v)\}$$

to be the set of nodes with routing table entries with endpoint u. Furthermore, let

$$FN_t(u) = \{v \in V \colon \exists (id_s, id(u), v_+, v_-) \in FT(v)\}$$

be the set of nodes that have a tunnel table entry with endpoint u. If p contains less than two tunnels, p has to contain a shortcut to a node in

$$Z(u) = \{u\} \cup NT_t(u) \cup ON_t(u) \cup FN_t(u) \cup \bigcup_{v \in ON_t(u)} FN_t(v).$$

On average, a node is the endpoint of

$$\mathbb{E}\left(|ON_t(u)|\right) = \mathbb{E}\left(|F_t(u)|/n\right) = \mathcal{O}\left(\log^\alpha n\right)$$

tunnels. By assumption, each tunnel is on average of length at most $\mathcal{O}(\log^r n)$, so that the number of tunnel table entries with endpoint v is

$$\mathbb{E}\left(|FN_t(v)|\right) = \mathcal{O}\left(\log^r n \, |ON_t(v)|\right) = \mathcal{O}\left(\log^{r+\alpha} n\right).$$

Hence

$$\begin{aligned} \mathbb{E}(Z_t) &= \mathbb{E}\left(|Z(u)|\right) \\ &\leq 1 + E\left(|NT_t(u)| + |ON_t(u)| \quad + |FN_t(u)| + \sum_{v \in ON_t(u)} |FN_t(v)|\right) \\ &= \mathcal{O}\left(|FN_t(u)| \, |ON_t(u)|\right) = \mathcal{O}\left(\log^{r+2\alpha-2} n\right). \end{aligned}$$

The number of nodes contacted during the tunnel discovery is at most $\mathbb{E}(M_t) = \log^\beta n$ by assumption, and the expected degree is constant. We apply Lemma 6.2 to determine the upper bound

$$P(H) = \mathcal{O}\left(\frac{\log^\beta n \log^\alpha n \log^r n \log^{r+2\alpha-2} n}{n}\right) = \mathcal{O}\left(\frac{polylog(n)}{n}\right).$$

Therefore, Equation 6.12 is dominated by the term q^2 for $q = \frac{1}{\log^k n}$, so that

$$P(NL_t^{\mathbf{T}} \leq \lambda_t(q)) = \mathcal{O}\left(q^2\right). \tag{6.13}$$

We can determine $\mathbb{E}(C_t - C_{t-1})$ in Equation 6.9 by Equations 6.11 and 6.13,

$$\mathbb{E}(C_t - C_{t-1}) = \Omega\left(1 - q^2 - 1 + \frac{q}{\log^r n}\right) = \Omega\left(\frac{q}{\log^{r+\beta} n} - q^2\right).$$

Now, we set $q = \frac{1}{\log^{\beta+r+1} n}$, so that $\mathbb{E}(C_t - C_{t-1}) = \Omega\left(\frac{1}{\log^{2r+2\beta+1} n}\right)$. By Equation 6.10, for all $t = \Omega(n \log^{2r+\beta+\alpha} n/q) = \Omega(n \log^{3r+\alpha+2\beta+1})$, we indeed have $\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \Omega\left(\log^r n\right)$. □

We have shown in Theorem 6.3 that for any $r > 0$, there exists t_r, such that $\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \Omega\left(\log^r n\right)$ for all $t > t_r$. Hence, for all $\epsilon > 0$ and $t = \Omega(n^{1+\epsilon})$, the expected mean tunnel length is $\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \omega\left(\log^r n\right)$ for all $r > 0$ and in particular not polylog.

6.3.3 Partially Determined Overlays

In this section, we consider *partially determined* virtual overlays, such that a link in the virtual overlay can potentially have several endpoints. For example, a virtual overlay based on Kademlia is partially determined because all nodes with a certain prefix are potential endpoints. First, we clearly define the class of partially determined overlays that are of interest. Second, we present the proof that polylog routing and stabilization complexity are mutually exclusive.

The routing complexity in a virtual overlay is only polylog if the number of tunnels (partially) contained in a route is polylog. Thus, we limit our analysis to such overlays. Overlays providing polylog routing share the characteristic that the probability of two nodes to be overlay neighbors generally decreases approximately anti-proportional to the number of closer nodes. Kleinberg showed in his seminal paper [86] that such a distance distribution allows for polylog routing in lattices whereas all other distance distributions are unable to provide such a performance. Consequently, we restrict our analysis to overlays with such a distance distribution.

In the following, we formalize the above idea such that it covers a large group of overlays. In order to simplify notation, we assume $|\mathbf{X}| = |V|$, i.e., the mapping id between nodes and coordinates is bijective. The results hold regardless of the above assumption. Let $B_d(v) = \{w \in V : \delta_X(v, w) \leq d\}$ denote the set of nodes within distance d of v. We say that a node u is a m-closest node of v if $u \in B_d(v)$ for some d such that $|B_d(v)| \leq m$, i.e., if there are at most m nodes with coordinates closer to $id(v)$ than u.

Definition 6.4. *We define a partially determined virtual overlay with a $1/B_d$ distance distribution by the following three conditions:*

1. *The set of tunnels between 2^{i+1}-closest but not 2^i-closest nodes makes up an asymptotically logarithmic fraction of all tunnels for $i = 0 \ldots \log n$, i.e., for all i, we have that*

$$E_i = \{p \in F \colon \exists d \colon e(p) \in B_d(s(p)) \setminus B_{d-1}(s(p)) \wedge 2^i < |B_d(s(p))| \leq 2^{i+1}\} \tag{6.14}$$

   is of size

$$|E_i| = \mathcal{O}\left(\frac{|F_t|}{\log n}\right). \tag{6.15}$$

2. *Furthermore, we assume that $|B_d(w)| = \theta(d^\mu)$ for some $\mu \in \mathbb{N}$, i.e., the coordinate space is essentially a μ-dimensional lattice.*

3. *When setting up a tunnel, the endpoint $e(p)$ is required to be a 2^{i+1}-closest but not within the 2^i-closest nodes to the startpoint for a fixed i.*

Condition 1 and 2 generalize Kleinberg's model such that it includes Kademlia among others. So, rather than defining a concrete probability for any two nodes to be neighbors, Condition 1) only bounds the number of neighbors within a certain distance. So, the distance distribution within each set E_i can be arbitrary for our proof. Similarly, Condition 2 abstracts from the lattice used in Kleinberg's model to metric spaces of bounded growth [36]. In this manner, the definition considers basically all overlays that achieve routing in a polylog number of virtual overlay hops, which is a necessary but not sufficient condition for achieving a polylog routing length in the F2F overlay. Before proving the main result, we state a decisive Lemma.

Lemma 6.5. *Let $O = (V, E, \mathbf{X}, id, F, \mathbf{R}, \mathbf{S})$ be a partially determined virtual overlay with a $1/B_d$ distance distribution. Consider a node $u \in V$. We say that a tunnel p improves by a factor $f = 2^{i-j}$ with $i > j+1$*

if $s(p)$ is a 2^i but not 2^{i-1}-closest node and $e(p)$ a 2^j but not 2^{j-1}-closest node of u. The probability of the improvement X to be at least f is

$$P(X \geq f) = \mathcal{O}\left(\frac{1}{f \log n}\right). \tag{6.16}$$

When applying Lemma 6.5 in the proof of Theorem 6.6, the node u corresponds to a potential endpoint of a newly constructed tunnel and the tunnel p is contained in the new tunnel. Thus, we apply Lemma 6.5 to provide a lower bound on both the number of contained tunnels as well as their length. The result has been shown for the one-dimensional case of Kleinberg's small-world model in [101], we here present a more general version.

Proof. The idea of the proof is to first show that any 2^j-closest node to u is not a 2^{i-1}-closest node to $s(p)$ and then use the property of the $1/B_d$ distance distribution.

More precisely, for $l = 0 \dots \log n$, let d_l be the smallest integer such that $|B_{d_l}(u)| \geq 2^l$. Furthermore, let pot_l be the set of 2^j-closest nodes v to u such that

$$pot_l = \{v \in B_{d_j}(u) : \exists d\colon v \in B_d(s(p)) \setminus B_{d-1}(s(p)) \wedge 2^l < |B_d(s(p))| \leq 2^{l+1}\},$$

i.e., a tunnel $\tilde{p}$ with startpoint u and endpoint $v \in pot_l$ would be an element of E_l defined in Equation 6.14 and v is a 2^{l+1}-closest but not 2^l-closest node to $s(p)$. Then, we can derive the desired probability by considering the fraction of potential endpoints for each l, i.e.,

$$P(X \geq f) = \sum_{l=1}^{\log n} \mathcal{O}\left(\frac{|pot_l|}{2^l \log n}\right). \tag{6.17}$$

The denominator follows from Equation 6.15 and the fact that there are about 2^l nodes v with $2^l < |B_{\delta_X(s(p),v)}(s(p))| \leq 2^{l+1}$. It remains to derive $|pot_l|$. For the proof of Equation 6.16, it suffices to show that $pot_l = 0$ for $l < i$. Then Equation 6.17 is maximized for $pot_{i-1} = 2^j$ and $pot_l = 0$ for $l > i$, i.e., if all 2^j-closest nodes are as close to $s(p)$ as possible. Hence, we now show that any 2^j-closest node to u is not a 2^{i-1}-closest node to $s(p)$.

The claim follows from the similarity of the coordinate space $\mathbf{X}$ to a μ-dimensional lattice that $d_i^\mu = \Omega\left(2^i\right)$, $d_j^\mu = \mathcal{O}\left(2^j\right)$, and

$$\delta_X(s(p), e(p)) = \Omega\left(\left(d_i^\mu - d_j^\mu\right)^{1/\mu}\right).$$

As a result, there exist at least $\Omega\left(d_i^\mu - d_j^\mu\right)$ nodes closer to $s(p)$ than $e(p)$. Because $i > j + 1$, we have $\Omega\left(d_i^\mu - d_j^\mu\right) = \Omega\left(2^{i-1}\right)$. So, there are indeed asymptotically at least 2^{i-1} nodes closer to $s(p)$ than any 2^j -closest node v to u. By this, we derive the claimed upper bound on the probability to improve by at least a factor $f = 2^{i-j}$

$$P(X \geq f) = \mathcal{O}\left(\frac{|pot_{i-1}|}{2^{i-1} \log n}\right) = \mathcal{O}\left(\frac{2^j}{2^{i-1} \log n}\right) = \mathcal{O}\left(\frac{2}{f \log n}\right).$$

□

With the help of Lemma 6.5, we now prove that partially determined virtual overlays with a $1/B_d$ distance distribution eventually fail to offer a polylog routing length.

Theorem 6.6. *Let $O_t = (V_t, E_t, \mathbf{X}, id, F_t, \mathbf{R}, \mathbf{S})$ be a virtual overlay with a $1/B_d$ distance distribution and tunnel discovery algorithm $\mathbf{T}$. For any $r \in \mathbb{N}_0$, there exists t such that*

$$\mathbb{E}\left(mean(L_t^{\mathbf{T}})\right) = \omega(\log^r n).$$

Proof. We focus on a set of tunnels with the minimal number of potential endpoints. Because they make up a logarithmic fraction of all tunnels due to the $1/B_d$ distance distribution, it suffices to show that their length exceeds $\omega(\log^{r+1} n)$ to show the claim. The main idea of the proof is to show that with high probability the newly constructed tunnel contains $\Omega\left(\frac{\log n}{\log \log n}\right)$ existing tunnels, each improving by at most 2^m for $m = k \log \log n$ to be determined during the proof. In other words, the fraction of nodes closer to potential endpoints decreases by at most a factor 2^m while following one tunnel. Based on this result, we then bound the expected tunnel length by $\omega(\log^r n)$ for $t \to \infty$.

We start by introducing some notation. Let $F_{t,i}$ denote the set of tunnels such that the endpoint is a $2^{(i+1)m}$-closest node but not a 2^{im}-closest node to the startpoint. We have $\mathbb{E}(|F_{t,i}|) = \theta\left(\frac{k \log \log n}{\log n} \mathbb{E}(|F_t|)\right)$

because O_t has a $1/B_d$ distance distribution (Condition 1). Denote the tunnel length distribution of $F_{t,i}$ by $L^{\mathbf{T}}_{t,i}$. In particular,

$$\mathbb{E}\left(mean(L^{\mathbf{T}}_t)\right) = \Omega\left(\frac{\log\log n}{\log n}\mathbb{E}\left(mean(L^{\mathbf{T}}_{t,0})\right)\right). \tag{6.18}$$

We only consider tunnels for which the number of potential endpoints U is at most $n^{0.25} = 2^{\log n/4}$. For this purpose, fix $i_0 = \lceil \frac{\log n}{4k\log\log n} \rceil$, the highest index i of interest. In the following, we divide the set $F_{t,i}$ into subsets $S_{t,i}$ and $S^{\perp}_{t,i}$. $S_{t,i} = S^0_{t,i} \cup S^1_{t,i} \cup S^2_{t,i}$ contains tunnels that are potentially shorter:

1. $S^0_{t,i}$: all remaining initially present tunnels,
2. $S^1_{t,i}$: newly constructed tunnels that contain a shortcut to any of the $n^{0.25}$-closest nodes to their endpoint
3. $S^2_{t,i}$: tunnels constructed at time $\tau < t$ which are not in $S^1_{t,i}$ and for which at least one of the contained tunnels is an element of $S_{\tau-1,j}$ for some $i_0 \geq j > i$, $0 \leq \tau < t$

The tunnel length distribution of tunnels in $S^{\perp}_{t,i}$ is denoted by $\Lambda^{\mathbf{T}}_{t,i}$ in the following.

Having introduced the necessary notation, we now give a more profound overview of the proof's essential steps. The actual proof is then rather technical, employing a variety of techniques from probability theory and calculus. We first show that $S^{\perp}_{t,0}$ makes up a non-negligible fraction of $F_{t,0}$ for $t \to \infty$. For this purpose, we derive a recursive formula of $\limsup_{t\to\infty}\mathbb{E}\left(|S_{t,i}|\right)$ and solve the recursion using the case $i = i_0$ as the recursion anchor. Secondly, we determine a bound on $\mathbb{E}\left(\Lambda^{\mathbf{T}}_{t,i}\right)$ for $i = 0$. We condition on the event that all contained tunnels improve by at most a factor 2^m and again derive a recursive relation expressing $\mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,i})\right)$ in terms of $\sum_{j=i+1}^{i_0}\mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,j})\right)$. In summary, we prove the claim by showing

$$\mathbb{E}\left(mean(L^{\mathbf{T}}_{t,0})\right) \geq \mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,0})\right)\mathbb{E}\left(\frac{|S^{\perp}_{t,0}|}{|F_{t,i}|}\right) = \Omega(\log^{r+1} n). \tag{6.19}$$

As for the proof of Theorem 6.3, we assume that $mean(L^{\mathbf{T}}_t) = \mathcal{O}\left(\log^r n\right)$ for all t to establish a contradiction.

In the first part, we prove that there exists t_A, so that for $t \geq t_A$,

$$\mathbb{E}\left(\frac{|S^{\perp}_{t,0}|}{|F_{t,0}|}\right) = \Omega(1).$$

For this purpose, we derive an upper bound on

$$\gamma_i = \limsup_{t\to\infty}\mathbb{E}\left(|S_{t,i}|\right).$$

Note that γ_i is well defined since $\frac{|S_{t,i}|}{|F_{t,i}|} \leq 1$.

We obtain the desired bound by expressing the probability to remove and to construct a tunnel in $S_{t,i}$ in terms of $\mathbb{E}\left(|S_{t,i}|\right)$. Let $E^R_{t,i}$ and $E^C_{t,i}$ denote the event that a tunnel in $F_{t,i}$ is removed and constructed, respectively, and p_t be the tunnel removed or constructed in step t. Since the probabilities of removal and construction are equal and the $1/B_d$ distance distribution is preserved, we have $P(E^R_{t,i}) = P(E^C_{t,i})$. Then the expected size of $S_{t,i}$ is recursively expressed as

$$\begin{aligned}\mathbb{E}\left(|S_{t,i}|\right) &= \mathbb{E}\left(|S_{t-1,i}|\right) + P(p_t \in S_{t,i}|E^C_{t,i})P(E^C_{t,i}) - P(p_t \in S_{t-1,i}|E^R_{t,i})P(E^R_{t,i}) \\ &= \mathbb{E}\left(|S_{t-1,i}|\right) + \left(P(p_t \in S_{t,i}|E^C_{t,i}) - P(p_t \in S_{t-1,i}|E^R_{t,i})\right)P(E^R_{t,i}),\end{aligned} \tag{6.20}$$

the expected size in the step before plus the expected change in size. By the definition of lim sup as the asymptotic upper bound, there exists a real number t_1, such that for all $t > t_1$ and all $i = 0, \ldots, i_0$,

$$\mathbb{E}\left(|S_{t,i}|\right) < \gamma_i + \frac{2}{n\,P(E^R_{t,i})}.$$

In particular, if $|\mathbb{E}\left(|S_{t,i}|\right) - \gamma_i| \leq 1/n$

$$P(p_t \in S_{t,i}|E^C_{t,i}) - P(p_t \in S_{t-1,i}|E^R_{t,i}) \leq \frac{1}{n} \tag{6.21}$$

applying Equation 6.20 for $t > t_1$. An upper bound γ_i can be derived as the maximal value of $|S_{t,i}|$, such that Equation 6.21 holds. By Equation 6.5, the probability of removing a tunnel in $S_{t-1,i}$ is bound by

$$P(p_t \in S_{t-1,i}|E^R_{t,i}) = \mathbb{E}\left(\frac{|S_{t-1,i}|}{\log^\beta n|F_{t-1,i}|mean(L^{\mathbf{T}}_{t,i})}\right) = \Omega\left(\frac{\mathbb{E}(|S_{t,i}|)}{n\log^{\alpha+\beta+r+1} n}\right). \tag{6.22}$$

For the construction, we consider the sets $S^0_{t,i}$, $S^1_{t,i}$, and $S^2_{t,i}$ individually. By definition, $S^0_{t,i}$ only consists of initially existing tunnels, so $P(p_t \in S^0_{t,i}|E^C_{t,i}) = 0$. For $S^1_{t,i}$, the probability that p_t contains a shortcut to a node within the $2^{i_0 k \log\log n}$-closest nodes to any potential endpoint is bound by Lemma 6.2. By assumption, upper bounds on the number of exchanged messages $\mathbb{E}(M^{\mathbf{S}}_t)$ for stabilization, the overall number of tunnels $\mathbb{E}(|F_t|)$, and the expected mean tunnel length $\mathbb{E}(mean(L^{\mathbf{T}}_t))$ are $\mathcal{O}(\log^\beta n)$, $\mathcal{O}(\log^\alpha n)$ and $\mathcal{O}(\log^{r+1} n)$, respectively. Recall that the number of potential endpoints $|U|$ for $i \leq i_0$ is bound by $n^{0.25}$. The number of potential target nodes Z_t is given by the size of the set of nodes that are $2^{i_0 k \log\log n}$-closest nodes to any node in U with upper bound

$$Z_t \leq 2^{i_0 k \log\log n}|U| = \mathcal{O}\left(2^{\lceil \frac{\log n}{4k\log\log n}\rceil k\log\log n} n^{0.25}\right) = \mathcal{O}\left(n^{0.5}\right),$$

so that by Lemma 6.2

$$P(p_t \in S^1_{t,i}|E^C_{t,i}) = \mathcal{O}\left(\frac{\log^\beta n \log^\alpha n \log^{r+1} n\, n^{0.5}}{n}\right) = \mathcal{O}\left(\frac{1}{n^{0.2}}\right). \tag{6.23}$$

The last step holds for n so large that $n^{0.3} \geq \log^{\beta+\alpha+r+1} n$. We assume that at most $\log^\beta n$ nodes are involved in the tunnel discovery on average, hence the new tunnel can consist of at most $\log^\beta n$ old tunnels. Furthermore, $\mathbb{E}(|F_{t,i}|) = \Omega\left(n\log^{\alpha-1} n\right)$. The probability that any of $\log^\beta n$ tunnels is an element of $S_{t,j}$ is obtained by a union bound

$$P(p_t \in S^2_{t,i}|E^C_{t,i}) = \mathcal{O}\left(\mathbb{E}(M^{\mathbf{S}}_t) n \max_{j:i_0\geq j>i}\left\{\frac{\mathbb{E}(|S_{t-1,j}|)}{n\log^{\alpha-1} n}\right\}\right) = \mathcal{O}\left(\log^\beta n \max_{j:i_0\geq j>i}\left\{\frac{\mathbb{E}(|S_{t-1,j}|)}{n\log^{\alpha-1} n}\right\}\right). \tag{6.24}$$

The fraction of initial tunnels converges to 0. So, there exists t_ϵ such that $P(S^0_{t,i}) \leq n^{-0.2}$ for all $i = 0,\ldots,i_0$ and $t > t_\epsilon$. It follows from Equations 6.23 and 6.24 that

$$P(p_t \in S_{t,i}|E^C_{t,i}) = \mathcal{O}\left(\frac{1}{n^{0.2}} + \max_{j:i_0\geq j>i}\left\{\frac{\mathbb{E}(|S_{t-1,j}|)}{n\log^{\alpha-\beta-1} n}\right\}\right).$$

By Equations 6.21 and 6.22, we have for the limit γ_i

$$\frac{1}{n^{0.2}} + \max_{j:i_0\geq j>i}\left\{\frac{\gamma_{i+1}}{n\log^{\alpha-\beta-1} n}\right\} - \frac{\gamma_i}{n\log^{\alpha+\beta+r+1} n} = \mathcal{O}\left(\frac{1}{n}\right).$$

The set $\{j : i_0 \geq j > i\}$ is empty for $i = i_0$. The upper bound on γ_{i_0} is hence

$$\gamma_{i_0} = \mathcal{O}\left(\frac{n\log^{\alpha+\beta+r+1} n}{n^{0.2}}\right) = \mathcal{O}\left(n^{0.8}\log^{\alpha+\beta+r+1} n\right).$$

For $i < i_0$, we obtain the recursive relation

$$\gamma_i = \mathcal{O}\left(\log^{2\beta+r+2} n\gamma_{i+1}\right).$$

So,

$$\begin{aligned}\gamma_0 &= \mathcal{O}\left((\log^{2\beta+r+2})^{\frac{\log n}{4k\log\log n}}\gamma_{i_0}\right) = \mathcal{O}\left(2^{(2\beta+r+2)\log\log n\frac{\log n}{4k\log\log n}}\gamma_{i_0}\right)\\ &= \mathcal{O}\left(n^{\frac{2\beta+r+2}{4k}}\gamma_{i_0}\right) = \mathcal{O}\left(n^{\frac{2\beta+r+2}{4k}} n^{0.8}\log^{\alpha+\beta+r+1} n\right)\end{aligned}$$

is an upper bound on γ_0. In order to show $\mathbb{E}\left(|S^\perp_{t,0}|/|F_{t,0}|\right) = \Omega(1)$, consider that $\mathbb{E}(|F_{t,0}|) = \Omega(n\log^{\alpha-1} n)$ and

$$\gamma_0 = \mathcal{O}\left(n^{(2\beta+r+2)/(4k)} n^{0.8}\log^{\alpha+\beta+r+1} n\right) = \mathcal{O}(n^{0.9}\log^{\alpha+\beta+r+1} n)$$

for $k \geq 2.5(2\beta + r + 2)$. Hence, there exists t_2 such that for $t > t_A = \max\{t_1, t_2\}$ indeed

$$\mathbb{E}\left(\frac{|S^{\perp}_{t,0}|}{|F_{t,0}|}\right) = \Omega\left(1 - \frac{\gamma_0}{\mathbb{E}(|F_{t,0}|)}\right) = \Omega(1) \tag{6.25}$$

because

$$\frac{\gamma_0}{\mathbb{E}(|F_{t,0}|)} = \mathcal{O}\left(\frac{n^{0.9}\log^{\alpha+\beta+r+1} n}{n\log^{\alpha-1} n}\right)$$

and $n\log^{\alpha-1} n$ dominates $n^{0.9}\log^{\alpha+\beta+r+1} n$ for n big enough. This completes the first part of the proof. We have shown that the set $S^{\perp}_{t,0}$ contains a non-negligible fraction of tunnels eventually. In the following, we show that these tunnels eventually exceed polylog length, hence the mean tunnel length exceeds polylog length.

In order to determine a lower bound on $\mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,i})\right)$ for t large enough, we determine a recursive relation for

$$\eta_i = \liminf_{t\to\infty} \mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,i})\right), i \leq i_0. \tag{6.26}$$

Trivially, $\eta_{i_0} = \Omega(1)$. Denote by H the event that the improvement is at most 2^m for all tunnels contained in the newly constructed tunnel p_t after the first node within the closest $n^{0.25}$ nodes to potential endpoints U has been reached. If H does not hold, the length of the new tunnel is at least 1, otherwise there is at least one tunnel from each set $F_{t,j}$ contained in p_t in order for the maximal improvement to be 2^m. Therefore, the expected length $\mathbb{E}(|p_t|)$ of a new tunnel p_t in $S^{\perp}_{t,i}$ is

$$\mathbb{E}\left(|p_t|\right) > 1 + P(H^{\perp}) \sum_{j=i+1}^{i_0} \mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,j})\right), \tag{6.27}$$

and hence asymptotically $\eta_i \geq 1 + P(H^{\perp})\sum_{j=i+1}^{i_0} \eta_j$. Now, we determine $P(H)$. We condition on the fact that tunnels in $S^{\perp}_{t,i}$ do not contain shortcuts, so tunnel table entries are not of interest. Furthermore, the probability that one tunnel improves by 2^m is $\mathcal{O}\left(\frac{1}{2^m \log n}\right)$ according to Lemma 6.5. Let Y_t denote the total number of routing tables entries considered during tunnel discovery. In expectation, the tunnel discovery algorithm $\mathbf{T}$ considers at most $\mathbb{E}(M^{\mathbf{S}}_t) = \mathcal{O}(\log^{\beta} n)$ nodes with on average $\log^{\alpha} n$ routing table entries, so $\mathbb{E}(Y_t) = \mathcal{O}(\log^{\beta+\alpha} n)$. Note that the function $g(y) = 1 - (1-a)^y$ for a constant $a \in (0,1)$ is concave as its second derivative is negative, hence by Jensen's Inequality $\mathbb{E}\left(g(Y_t)\right) \leq g\left(\mathbb{E}(Y_t)\right)$. The probability of an improvement by at least a factor 2^m for $m = k\log\log n$ is hence

$$P(H^{\perp}) = \mathcal{O}\left(\mathbb{E}\left(1 - \left(1 - \frac{1}{2^m \log n}\right)^{Y_t}\right)\right) = \mathcal{O}\left(1 - \left(1 - \frac{1}{2^m \log n}\right)^{\log^{\alpha+\beta} n}\right) = \mathcal{O}\left(\frac{\log^{\alpha+\beta} n}{\log^{k+1} n}\right).$$

Thus, $P(H^{\perp}) \geq 1/2$ for any $k > \alpha + \beta - 1$ and n large enough. We get for η_i that

$$\begin{aligned}\eta_i &\geq 1 + 1/2 \sum_{j=i+1}^{i_0} \eta_j \geq 1 + 1/2 \sum_{j=i+2}^{i_0} \eta_j + 1/2\eta_{i+1} \\ &\geq 1 + \eta_{i+1} - 1 + 1/2\eta_{i+1} = 1.5\eta_{i+1}.\end{aligned} \tag{6.28}$$

By Equation 6.28, η_0 is recursively determined as

$$\begin{aligned}\eta_0 &= \Omega\left(1.5^{\log n/(4k\log\log n)}\right) = \Omega\left(2^{\log 1.5 \log n/(4k\log\log n)}\right) \\ &= \omega\left(2^{(r+1)\log\log n}\right) = \omega\left(\log^{r+1} n\right)\end{aligned}$$

and hence for $t > t_B$ for some t_B, $\mathbb{E}\left(mean(\Lambda^{\mathbf{T}}_{t,0})\right) = \omega\left(\log^{r+1} n\right)$. Setting $k > \max\{2.5(2\beta+r+2), \alpha+\beta-1\}$, the expected mean tunnel length in $F_{t,0}$ is at least $\omega(\log^{r+1} n)$ by Equation 6.19 for $t \geq \max\{t_A, t_B\}$. The claim $\mathbb{E}\left(mean(L^{\mathbf{T}}_t)\right) = \omega(\log^r n)$ follows from Equation 6.18. □

In this section, we have established the asymptotic impossibility of efficiently maintaining virtual overlays in their common form. However, it remains to set these asymptotic bounds in relation to concrete results with regard to both the speed and degree of tunnel length increase for selected systems.

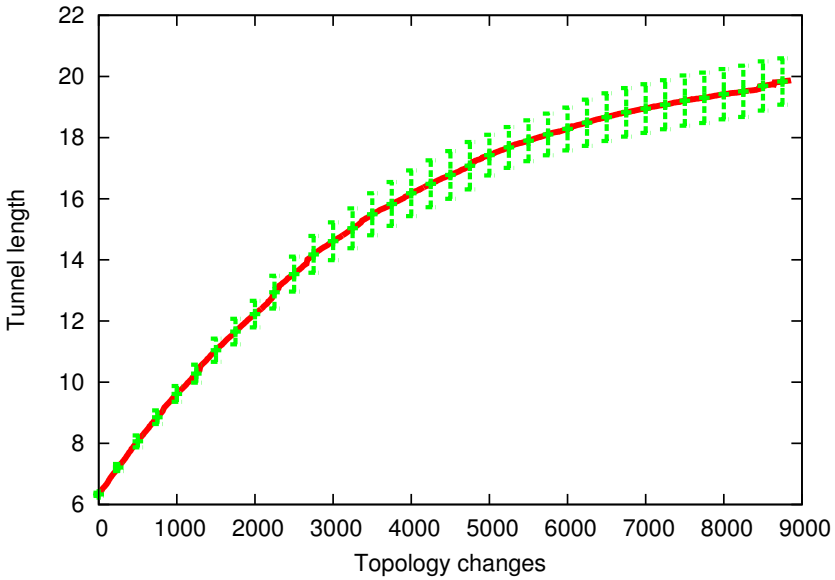


Figure 6.6: Mean tunnel length, ∼12k online nodes

6.4 Simulation

Since our theoretical results are of an asymptotic nature, we performed a simplified simulation study to assess the behavior of virtual overlays. We deliberately chose idealized conditions and simplified churn to highlight the extent of the problems virtual overlay approaches are facing, even at moderate network sizes.

For the experiment, we integrate a Chord-like virtual overlay in our simulation model as follows: using a b-bit coordinate space, each node establishes tunnels to their predecessor and successor, as well as to the nodes with coordinates succeeding $id(v) + 2^i \ mod \ 2^b$ for $i = 1 \dots b-1$. The tunnel discovery was implemented as a greedy routing through the virtual overlay, along existing tunnels or direct neighbors, towards the destination coordinate, chosen to complete the routing table.

We reconstruct each disrupted tunnel at its initial node instead of the last hop prior to the departing node. This allows for the discovery of new short tunnels, as opposed to the simple stitching of existing tunnels. Disruption of the ring, when a set of nodes that connected two components of the network failed, and re-connection of disconnected components were handled according to [40]: Each ring is identified, known to all nodes participating in it, and upon discovery of a "superior" ring (with a lower ID, by definition), nodes release tunnels to their previous neighbors, informing them of the new ring ID, and establish tunnels to the neighbors in the newly joined ring.

To approximate realistic assumptions, we focused *FB* as underlying trust graph to connect the nodes, since our recent measurement study [134] indicate that this reflects the size of the current Freenet deployment well. Furthermore, the low size of the largest component in *WOT* and *SPI* under churn did not encourage meaningful results. The churn events were generated according to the churn data introduced in Section 4.1.3. All results are averaged over 15 simulation runs, and presented with 95%-confidence intervals computed using the student-t distribution.

The results denote a steep incline of the length of tunnels (which each represent a single hop in the end-to-end overlay routing), even within the first 10.000 churn events. This corresponds to about ten minutes of Freenet churn traces and represents the arrival or departure of only a fraction of the entire node set (cmp. Fig. 6.6). While the increase subsequently slows down, the experiments indicate that the asymptotic results apply already to networks of rather small size, and that virtual overlay approaches cannot provide efficient routing even under idealized conditions, and suffer from degraded routing already after a very short period of operation.

6.5 Discussion

In this chapter, we have shown that polylog stabilization and polylog routing complexity are mutually exclusive in virtual overlays without an underlying routing protocol. So, they could be applied for use cases with a low dynamic, i.e., under the assumption that the overlay consists of servers or desktop computers that are online for most of the time. Then, an expensive stabilization algorithm does not have a large impact on the overall communication complexity. However, in a dynamic overlay including

possibly even mobile users, virtual overlays following the state-of-the-art approaches inherently fail to satisfy our requirements.

Recall from our state-of-the-art analysis in Chapter 3 that we identified virtual overlays as the only promising state-of-the-art approach. However, greedy embeddings offer highly efficient routing without altering the topology but have not been considered in the context of F2F overlays. We thus focus on greedy embeddings in the following.

Chapter 7

Greedy Embeddings

In this chapter, we evaluate greedy embeddings with regard to our requirements. More precisely, we check if there are requirements tree-based greedy embeddings inherently fail to achieve simultaneously. In terms of the requirements defined in Section 2.6, we focus on the requirement of balanced content addressing and its relation to stabilization complexity and censorship-resistance.

Recall from Section 2.2 that network embeddings facilitate efficient routing between pairs of nodes. For this purpose, greedy embeddings establish a spanning tree and assign coordinates to nodes based on their position in the tree. When designing a protocol for content addressing based on these tree coordinates, we have to deal with the fact that a tree is a hierarchical structure and spanning trees tend to be unbalanced. More precisely, it is unclear how to address content such that nodes at different levels and with a diverse number of descendants receive roughly the same amount of content. The issue of assigning content in a balanced fashion is further complicated due to the requirement of hiding the exact topology. Indeed, we show that guaranteeing balanced content addressing requires either exhaustive stabilization or enables an adversary to execute censorship.

We first present our results in an informal manner in Section 7.1, followed by a formal model in Section 7.2 and the subsequent theoretical analysis in Section 7.3. Afterwards, in Section 7.4, we introduce several exemplary algorithms for content addressing in tree-based embeddings. For these content addressing algorithms, we then perform a simulation study to validate the asymptotic bounds in Section 7.5. In Section 7.6, we conclude and discuss the impact of the results on the remainder of the thesis. Our results have been partially published in [139].

7.1 Statement of Result and Proof Idea

In this section, we first sketch our proof that balanced content addressing requires information about the spanning tree structure. Recall that we say that the content addressing is balanced if the fraction of content per node is distributed roughly uniformly with the maximal fraction of content per node being $\mathcal{O}\left(\frac{\log n}{n}\right)$. After proving that balanced content addressing requires that the number of descendants per node is (approximately) known, we explain that such information cannot be reliably obtained without a secure consensus protocol. We thus deduce that a secure consensus protocol is a necessary but not necessarily sufficient condition for balanced and censorship-resistant content addressing. Last, we consider the communication complexity of such consensus protocols and find that they are highly inefficient. In summary, greedy embeddings fail to provide balanced content addressing, censorship-resistance, and efficient stabilization concurrently.

A spanning tree is a hierarchical structure, i.e., nodes are grouped according to their level in the tree. It is not immediately clear how to provide content addressing such that nodes on different levels receive roughly the same fraction of content. Moreover, spanning trees are likely to be unbalanced, depending on the construction algorithm and the nature of the graph. In other words, leaves can be at different levels and even the number of descendants of nodes on the same level varies.

Now, consider content addressing based on a spanning tree. Let u be a parent of multiple children and consider the subtrees rooted at these children. A straight-forward algorithm disregarding the actual topology would assign roughly the same amount of content to each subtree, i.e., the combined fraction of content assigned to all nodes in a subtree should be approximately equal for all subtrees at the same level. In this manner, subtrees with less nodes receive a disproportional high fraction of content and hence individual nodes are overloaded, as illustrated on the left of Figure 7.1.

Hence, assigning the same fraction of content to all subtrees at the same level is insufficient for unbalanced spanning trees. If the topology of the tree is known, the addressing can be biased such

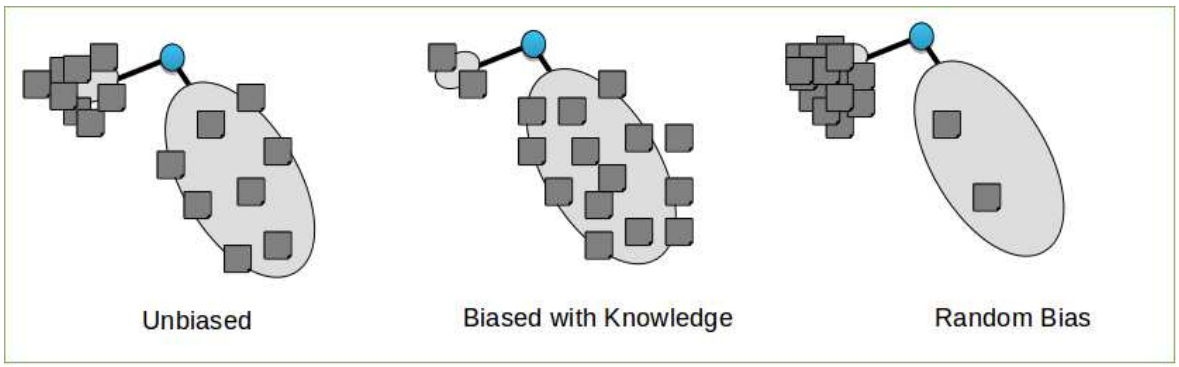


Figure 7.1: Tree-based greedy embeddings and content addressing (dark squares indicating content): Without biasing the fraction of content per subtree, the addressing is likely to be unbalanced (left), including topology knowledge in the assignment of coordinates can achieve balanced content addressing but allows attacks (middle), bias without knowledge is likely to increase imbalance (right) as smaller subtree might receive more content

that the fraction per subtree roughly corresponds to the fraction of nodes in that subtree. In contrast, without considering the topology, biasing the fraction of content per subtree is likely to lead to a higher imbalance. Both scenarios are illustrated in Figure 7.1. Intuitively, it follows that balanced content addressing requires information about the number of nodes per subtree.

In Section 7.3, we formalize the intuition that balanced content addressing requires structural knowledge when constructing the embedding. The idea of the proof is to show that any embedding algorithm will inevitably assign a constant fraction of content to a subtree consisting of one node for some graph. More precisely, we show that without knowledge of the subtree size, any embedding algorithm can entail an arbitrary imbalance in the fraction of assigned content, i.e., the maximal fraction of content assigned to one node scales with $\Omega(1)$ rather than $\mathcal{O}\left(\frac{\log}{n}\right)$. Under the assumption that the total amount of content increases linearly with the number of nodes, a few nodes in large-scale overlays are responsible for an enormous amount of content. Clearly, these nodes cannot deal with such enormous load and are likely to delete most of the content. As a consequence, the overlay would be unable to provide reliable content sharing, one of our two main functionalities.

Now, the above result shows that balanced content addressing requires (an approximation of) the number of nodes in each subtree. In the following, we consider if we can provide such information and still achieve our remaining goals, in particular resistance to attacks and efficient stabilization.

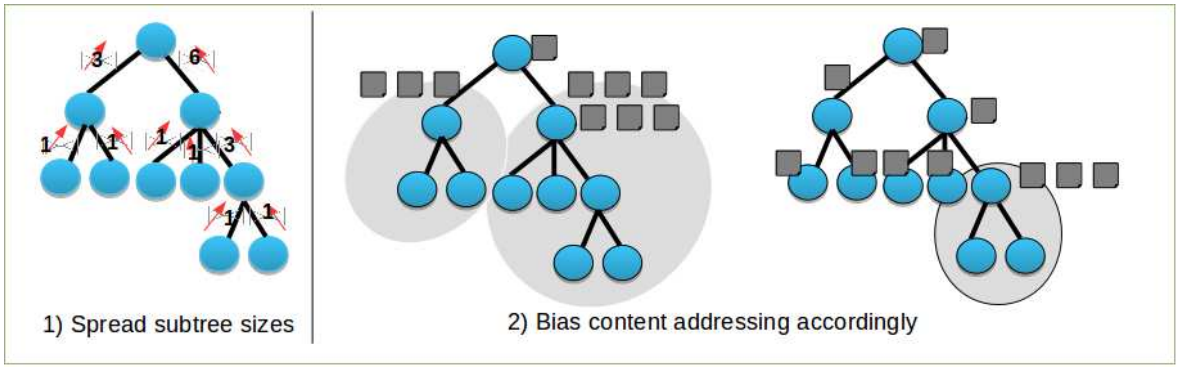


Figure 7.2: Straight-forward approach for balanced content addressing in tree-based embeddings: In the first phase, nodes inform their parents of the number of nodes in their subtree, starting from the leaves up to the root. In the second phase, internal nodes, starting from the root, assign the coordinates of their children such that the expected fraction of content stored at nodes in each subtree is roughly proportional to the number of nodes in that subtree.

In order to exemplary illustrate the problem, we start by presenting a straight-forward algorithm for spreading the subtree sizes and assigning coordinates accordingly, displayed in Figure 7.2. The algorithm first counts the number of descendants per subtree from the leaves to the root. The root assigns its children coordinates such that the expected fraction of content mapped to nodes in the subtrees corresponds to the fraction of nodes in the subtree. Subsequently, parent nodes u assign child coordinates in the same manner, i.e., such that the fraction of the subtree rooted at u is divided upon the subtrees rooted at

children of u in proportion to their size. An advantage of the algorithm is that nodes only know the size of subtrees rooted at their children but not the complete structure of the spanning tree. However, nodes can lie about the size of their subtree in order to be assigned a large fraction of content, which they can then censor. In particular, a malicious root node can sabotage the above protocol by assigning the majority of content to a subtree that is either non-existent or consists of Sybil nodes. Thus, a necessary but not necessarily sufficient condition for the above algorithm to be resilient is a secure root election protocol, which is essentially a secure distributed consensus protocol.

The above example gives an intuition on why balanced content addressing enables censorship unless additional protection schemes are applied. Now, we argue that a distributed consensus protocol is always required in order to prevent censorship, not only for the above exemplary algorithm. In general, it is possible to decide on the fraction of content assigned to each subtree by i) distributed consensus involving all nodes or ii) selecting a subset of nodes to make the decision. In the latter case, we need a distributed consensus protocol for the selection of the subset. Furthermore, as soon as one of the selected nodes leaves, the nodes have to elect a new set. Thus, we always need a distributed consensus protocol and we have to apply it repeatedly.

From the above argument, we can now determine the impact of the distributed consensus protocol on the stabilization complexity. It is necessary to apply the protocol whenever a node involved in the decision departs, thus with at least probability $\Omega\left(\frac{1}{n}\right)$ per topology change. The lower bound of $\Omega\left(\frac{1}{n}\right)$ corresponds to the scenario that the coordinate assignment decision depends on one node, for example the root. Distributed consensus protocols such as [98] require each node to communicate with all others, hence their complexity scales with $\Omega(n^2)$. As a consequence, preventing attacks requires at least an expected stabilization complexity of $\Omega(n)$ and is thus inefficient.

So, balanced content addressing in tree-based embeddings is only possible at the price of increasing the vulnerability to attacks or counter-acting such attacks through high stabilization complexity. We now formally prove that indeed balanced content addressing requires the size of the subtrees and give some empirical evidence of the fact.

7.2 Notation and System Model

In this section, we introduce basic notation and formally express our goals. The key terms we need to define are those of a (greedy) embedding, a content addressable storage, and a stabilization algorithm for such a structure.

7.2.1 Graphs and Embeddings

Recall from Section 2.2 that an embedding $id : V \to \mathbf{X}$ assigns each node of a graph $G = (V, E)$ a coordinate from a metric space $(\mathbf{X}, \delta_X)$. The embedding is called greedy if the standard greedy routing algorithm always succeeds in finding a path between any two nodes in the same connected component of G. Such a path is characterized as being *monotonous* in the sense that the coordinate of any node on the path is closer to the destination coordinate than the coordinate of the previous node on the path.

All known greedy embedding algorithms rely on spanning trees for the construction of the embedding. Thus, the embedding usually consists of the following four steps (which can partially be executed in parallel):

1. Construct a spanning tree T_G of the graph G.
2. Each internal node of T_G enumerates its children.
3. The root node is assigned a specific coordinate.
4. Iteratively, children are assigned coordinates based on their parent's coordinates and the enumeration index from step 2.

We call an embedding id constructed as above a *tree-based embedding* or *tree-based greedy embedding* if id is greedy. Greedy embeddings allow the discovery of a node by a standard greedy algorithm. However, there is little work on how to store and retrieve content in such an embedding.

7.2.2 Balanced Content Addressing

Content addressing generally refers to assignment *keys* or *addresses* to content in order to efficiently locate the content. In the context of distributed systems, content addressing specifically refers to the mapping of content to nodes based on node coordinates and content addresses sharing the same metric

space. Here, we assume that files are mapped to the node with the closest coordinate to the file's key. All results presented in this chapter remain valid if content is stored on $k > 1$ nodes, using e.g., the k closest nodes to one key or k distinct keys per file.

We assume that the routing algorithms $\mathbf{R}_{node}$ and $\mathbf{R}_{content}$ corresponds to the standard greedy routing protocol. So, each node u forwards a request for an address add to the neighbor v with the closest coordinate to the requested address if $\delta_X(id(v), add) < \delta_X(id(u), add)$. Otherwise, the routing terminates at u.

For brevity, we usually equate a node's coordinate with the node itself. So, we refer to the node with the closest coordinate to an address as the closest node. Similar, we say that two nodes have a certain distance rather than saying that the coordinates of the nodes have a certain distance. While not being perfectly accurate, the abbreviation considerably increases the readability.

Greedy embeddings guarantee that $\mathbf{R}_{node}(s, id(e))$ terminates at node e starting from any source node s. In order to distribute and retrieve content in a greedy embedding, we furthermore require that $\mathbf{R}_{content}(s, x)$ terminates at the node with the closest coordinate to x for all $x \in \mathbf{X}$. Thus, we slightly extend the definition of a greedy embedding.

Definition 7.1. *Let $id : V \to \mathbf{X}$ be a greedy embedding. For any coordinate $x \in X$, we denote the set of nodes with the closest coordinates to x by $V(x) = argmin_{v \in V}\{\delta_X(id(v), x)\}$. Then id is called a* content addressable greedy *embedding if for all $x \in \mathbf{X}$*

1. *$\mathbf{R}_{content}(x)$ terminates at a node $v \in V(x)$, and*
2. *Any two distinct nodes $v, u \in V(x)$ are in a parent-child relationship.*

The second condition in Definition 7.1 is necessary to guarantee the retrieval of content that is closest to multiple nodes. If that is the case, Condition 1 ensures that the content is stored at one such node v, whereas Condition 2 ensures that it can be discovered even if $\mathbf{R}_{content}(s, x)$ does terminate at a different node u. If indeed $\mathbf{R}_{content}(s, x)$ terminates at u, u can forward the message either to its children or its parent to retrieve a content with key x. Note that the above guarantees only hold in the absence of failures or intentional manipulation of the routing or embedding protocol. Now, we can formally define the notion of using a distributed system as a content addressable storage.

Definition 7.2. *A* content addressable storage *is defined by a tuple $CAS = (G, \mathbf{X}, id, \mu)$ such that $G = (V, E)$ is a graph with a content addressable greedy embedding $id : V \to \mathbf{X}$, and μ is a probability mass on $\mathbf{X}$ defining the likelihood of coordinates to be keys.*

For example, the keys of files can be computed using cryptographically secure hash functions that assign each file a b-bit key approximately uniformly at random. As a consequence, the coordinate space $\mathbf{X}$ corresponds to the set of all b-bits numbers and the probability measure μ is approximately the point measure, i.e., each coordinate is assigned the same probability. However, we require the generality of Definition 7.2 to show the universal impossibility of balanced content addressing in greedy embeddings without the revelation of topology information.

The above definition does not demand that the content is distributed on the nodes in a balanced manner. Thus, we now characterize the notion of balanced or fair content addressing.

Definition 7.3. *Let $CAS = (G, \mathbf{X}, id, \mu)$ be a content addressable storage. Furthermore, let $\mathbb{B}(v) = \{x \in \mathbf{X} : \forall w \in V : \delta_X(id(v), x) \leq \delta_X(id(w), x)\}$ be the set of coordinates x in $\mathbf{X}$ so that $id(v)$ is (one of) the closest node coordinate(s). CAS is said to be* f-balanced *for a real-valued factor $f \geq 1$ if*

$$\forall v \in V : \mu(\mathbb{B}(v)) \leq f \cdot \frac{1}{|V|}. \tag{7.1}$$

An embedding algorithm $\mathbf{A}$ is called f-balanced if it is guaranteed to generate embeddings id such that any content addressable storage $(G, \mathbf{X}, id, \mu)$ is f-balanced.

In other words, Definition 7.3 states that the expected fraction of content assigned to any node should be at most f times the average amount of content per node. We require that any content addressable storage is $\mathcal{O}(\log n)$-balanced, as motivated in Section 2.6.

Before starting our actual analysis, we consider the usability of the above definition in an overlay with heterogeneous nodes. Definition 7.3 is based on the assumption that balanced content addressing is advantageous for a well performing system. As motivated in Section 2.6, achieving a uniform distribution of content is a sensible goal despite the fact that the storage capacities of individual nodes might differ drastically. Nodes can the always participate as multiple identities in order to balance the content of nodes according to their resources [73]. Thus, we focus on the possibility of achieving a uniform distribution of content on nodes.

7.3 Analysis

In this section, we show that greedy embeddings inherently cannot provide balanced content addressing without revealing the number of descendants of a node in the spanning tree. It follows that greedy embeddings cannot simultaneously provide balanced content addressing, resilience, and efficient stabilization. Before showing the main result, we first formally ascertain that indeed the second closest node to an element $x \in \mathbf{X}$ in any greedy tree-based embedding is the parent of the closest node, as illustrated in Figure 7.3. Based on this fact, we can show that coordinates that are normally mapped to descendants of a node u are mapped to u in the absence of these descendants. Thus, largely simplifying the result in Theorem 7.5 in order to illustrate its underlying idea, leaves at a low depth of the spanning tree are usually responsible for a large fraction of coordinates and thus content.

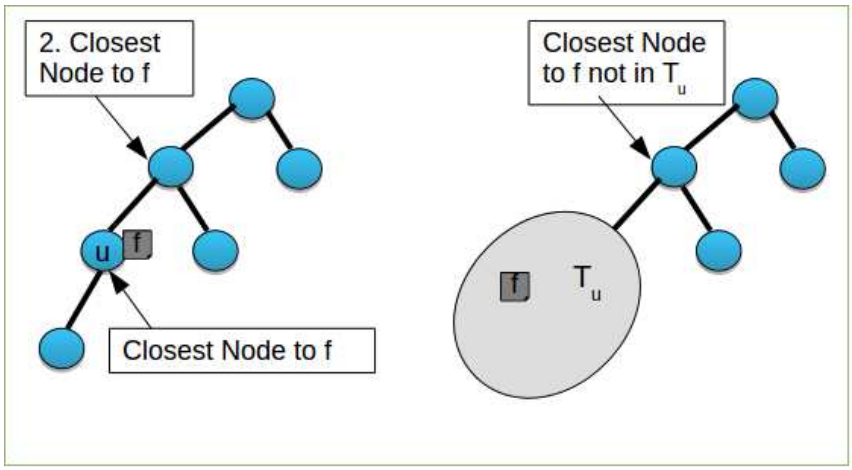


Figure 7.3: Illustration of Theorem 7.4: For greedy routing to work, the second closest node to an address has to be the parent of the closest node to that address (left). More generally, consider any address closest to a node in certain subtree. Then the closest node that is not in the subtree is the parent of the subtree's root (right).

Lemma 7.4. *Let $CAS = (G, \mathbf{X}, id, \mu)$ be a content addressable storage with a tree-based embedding $id : V \to \mathbf{X}$ for a spanning tree T of G. Consider a subtree $T_u = (V_u, E_u)$ rooted at node u. For any $x \in \cup_{v \in V_u} \mathbb{B}(v)$, i.e., any element in x closest to a node in T_u, we have*

$$argmin_{w \in V \setminus V_u} \{\delta_X(id(w), x)\} = \{parent(u)\},$$

i.e., the closet node to x not contained in V_u is the parent of T_u's root u.

Proof. The result is a direct consequence of Definition 7.1 of a generalized greedy embedding. We here shortly present the proof by contradiction. Assume the graph G is a tree and that there are nodes $w \in V \setminus (V_u \cup \{parent(u)\})$ such that $\delta_X(id(w), x) \leq \delta_X(id(parent(u)), x)$. We show that if that is the case, CAS cannot satisfy the requirements of a content addressable embedding. Let w_x be the node in $V \setminus (V_u \cup \{parent(u)\})$ closest to x. As w_x does not have any neighbor in T that is contained in V_u, w_x does not have any neighbor closer to x than itself. Therefore, the routing algorithm $\mathbf{R}_{content}(w_x, x)$ terminates at w_x. As a consequence, we need to have $w_x \in V(x)$ for id to satisfy the convergence to a closest node required by Condition 1 in Definition 7.1. However, this contradicts Condition 2 stating that w_x should be in parent-child with any closest node to x. Thus, if w_x exist, CAS is not a content addressable embedding. □

Now, we prove our main result. We show that any tree-based embedding algorithm $\mathbf{A}$ is $\Omega(n)$-balanced, i.e., for any coordinate space $\mathbf{X}$ and probability measure μ, there exists a graph G such that the content addressable storage $CAS = (G, \mathbf{X}, id, \mu)$ with id generated by $\mathbf{A}$ is $\Omega(n)$-balanced.

Theorem 7.5. *Let $\mathbf{A}$ be a tree-based greedy embedding algorithm oblivious to the structure of the spanning tree. Then $\mathbf{A}$ is $\Omega(n)$-balanced.*

Proof. For the proof, we provide an exemplary class of graphs that require $\mathbf{A}$ to assign at least a constant fraction of addresses to the same node. Recall from Definition 7.3 that $\mu(\mathbb{B}(v)$ denotes the probability that an address is assigned to v. Note that $\mu(\mathbb{B}(v)$ is equal to the expected fraction of content assigned to v. So, in the light of Lemma 7.4, $\sum_{v \in V_u} \mu(\mathbb{B}(v))$, the expected fraction of content assigned to nodes in a

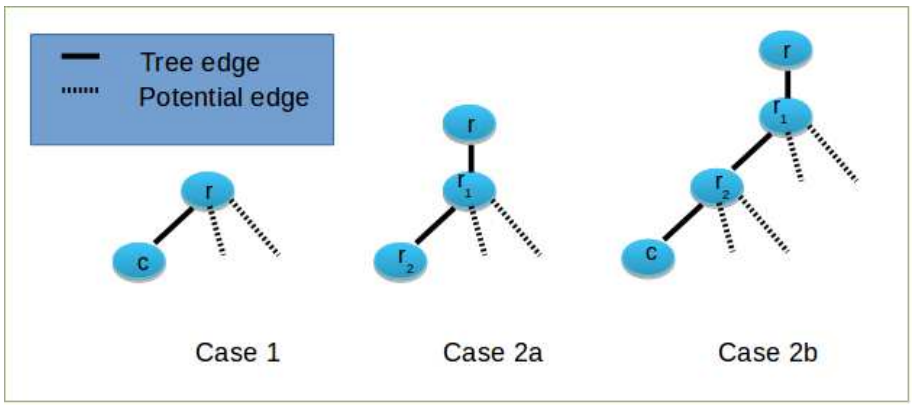


Figure 7.4: Adverse scenarios for balanced content addressing, illustrating the proof of Theorem 7.5: majority of content stored on displayed nodes only, independent of total number of nodes.

subtree $T_u = (V_u, E_u)$ rooted at a node u, is independent of V_u's topology. Furthermore, the enumeration of children in the tree during the embedding is independent of the number of descendants of these children. Hence, for each child c of a node v, there exists an enumeration of children that the subtree rooted at c is assigned the most addresses of all subtrees rooted at children of v. Formally, let u be a node with children $children(u)$ and define

$$\eta(u) = \sum_{w \in children(u)} \sum_{v \in V_w} \mu(\mathbb{B}(v))$$

as the expected fraction of content assigned to u's descendants. Then there exists an enumeration such that the content assigned to the subtree rooted at c is the largest and thus at least the average

$$\eta(c) + \mu(\mathbb{B}(c)) = \sum_{v \in V_c} \mu(\mathbb{B}(v)) \geq \frac{\eta(u)}{|children(u)|}. \tag{7.2}$$

Equation 7.2 allows us to define graphs G such that $CAS = (G, \mathbf{X}, id, \mu)$ is $\Omega(n)$-balanced. Let G be a tree of maximal degree K such that each node has either only one neighbor or at least one neighbor with degree 1. In the first step of the embedding, $\mathbf{A}$ determines a root node. Afterwards, the spanning tree is established. Due to the obliviousness of $\mathbf{A}$ to the graph structure, each subtree rooted at a node u is assigned an expected fraction of content independently of its size. For instance, if all subtrees receive roughly the same fraction of expected content, the smallest subtree receives the same fraction of content as all others. If the fraction of content per subtree is randomly biased, there are graphs for which the largest fraction of content is assigned to the smallest subtree. Recall Figure 7.1 for an illustration of the different scenarios.

We distinguish two cases concerning the degree of the selected root. Either $\mathbf{A}$ selects a root node r of degree at least 2 or of degree 1. In the following, we consider both cases and show that regardless of the degree of the root, one node is assigned a constant fraction of content in expectation.

If r is a node of degree at least 2, r has one child c with degree 1 and at most $K > 1$ children in total. The scenario is displayed on the left side of Figure 7.4. Because c has no descendants, we have $\eta(c) = 0$. If $\eta(c) = 0$, by Equation 7.2, the children of r can be enumerated so that

$$\mu(\mathbb{B}(c)) \geq \frac{\eta(r)}{|children(r)|} = \frac{1 - \mu(\mathbb{B}(r))}{K}, \tag{7.3}$$

because the coordinates not assigned to the root r have a mass of $1-\mu(\mathbb{B}(r))$ and at least $1/K$ are assigned to the subtree rooted at c. It follows from Equation 7.3 that either $\mu(\mathbb{B}(r)) \geq 0.5$ or $\mu(\mathbb{B}(c)) \geq \frac{1}{2K}$. As a consequence, CAS with the embedding provided by $\mathbf{A}$ is $\Omega\left(\frac{n}{2K}\right)$-balanced.

If now r is a node of degree 1, it has only one child r_1. On the second level of the tree are at most $K-1$ nodes, the remaining neighbors of r_1. As a consequence, at least one subtree rooted at a second level node r_2 is assigned more than $1/(K-1)$-th of the content not assigned to r or r_1

$$\mu(\mathbb{B}(r_2)) + \eta(r_2) \geq \frac{1 - \mu(\mathbb{B}(r)) - \mu(\mathbb{B}(r_1))}{K-1}. \tag{7.4}$$

If r_2 is of degree 1 and thus has no descendants, as displayed in the middle of Figure 7.4, we have $\eta(r_2) = 0$. As a result from Equation 7.4, there exists a node $v \in \{r, r_1, r_2\}$ such that $\mu(\mathbb{B}(v)) \geq \frac{1}{3(K-1)}$.

Otherwise, if r_2 is of degree at least 2, r_2 has one child c of degree 1 and at most $K-1$ children in total, corresponding to the scenario on the right of Figure 7.4. As above, there exists an enumeration such that $\mu(c) \geq \frac{\eta(r_2)}{K-1}$. Hence, again by Equation 7.4, we either have $\mu(\mathbb{B}(v)) \geq \frac{1}{6(K-1)}$ for one $v \in \{r, r_1, r_2\}$ or $\mu(\mathbb{B}(c)) \geq \frac{1}{2(K-1)^2}$. As a consequence, CAS with the embedding id provided by $\mathbf{A}$ is $\Omega\left(\frac{n}{2(K-1)^2}\right)$-balanced.

In any case, $\mathbf{A}$ embeds the considered graph G such that $(G, \mathbf{X}, id, \mu)$ is f-balanced with a linearly increasing f. So, the algorithm $\mathbf{A}$ is indeed $\Omega(n)$-balanced. □

Theorem 7.5 ascertains the existence of graphs that cannot be embedded such that the content addressing is balanced without revealing sensitive information. Clearly, there are also graphs that can be embedded in such a manner, namely graphs with balanced spanning trees. Hence, the above result only states that it is impossible to provide any guarantees that F2F topologies can be embedded in a balanced fashion without making further assumptions about their structure. Nevertheless, intuitively, social networks should result in unbalanced trees due to their scale-free degree distribution. The high fraction of low degree nodes are bound to result in leaves at a high level, which are then likely to be assigned a large fraction of content. To fortify the above intuition, we evaluated the load balancing of two embedding algorithms in a simulation study on real-world social networks.

7.4 Algorithm Design

In this section, we present two algorithms for content addressing in spanning trees. In previous work [82], we proposed a method for content addressing in the PIE embedding [80], which we present in Sections 7.4.1 and 7.4.2. As [82] maps all content to nodes with less than two children, we present a novel modification with the assigning content to all nodes in Section 7.4.3. Last, we show that the presented embedding algorithms are indeed greedy and quantify the fraction of content assigned to a node in terms of its level and number of children.

In addition to the presented algorithms, we developed an algorithm for balanced content addressing in tree-based embeddings. Due to its vulnerability to malicious nodes, the algorithm cannot be applied for F2F networks but is of interest for e.g., embedding AS topologies in content-centric networking [139]. We include it in Appendix D.

7.4.1 Unweighted PIE Embedding

In the following, we present an unweighted variant of PIE. While the original algorithm requires a coordinate length of $\mathcal{O}(\log^3 n)$ for a graph with a logarithmic diameter due to encoding of edges weights, the unweighted variant only requires coordinates of length $\mathcal{O}(\log^2 n)$.

Before starting the actual description, note that the *least common ancestor* of two nodes u and v is the ancestor w, i.e., a node w on the path between a node and the root, such that the level of w is the highest of all common ancestor of u and v. In other words, w is the first common ancestor reached when traversing a path from u or v to the root and it is the only ancestor that is on the shortest path between the two nodes in the tree. For this reason, w is particularly important when deriving bounds on the routing complexity.

In the following, we present the embedding algorithm PIE for unweighted graphs. During the spanning tree construction, each node enumerates its children. A greedy embedding can then be achieved by assigning a vector of length 0 to the root, and iteratively assigning each child the parent coordinate concatenated with an additional element corresponding to the index in the parent's enumeration. An example is presented on the right of Figure 7.5. Now, let $|vec|$ denote the length of a vector and $cpl(vec_1, vec_2)$ the common prefix length of two vectors, i.e., the number of leading elements the vectors vec_1 and vec_2 have in common. So, the common prefix length of the coordinates $id(u)$ and $id(v)$ corresponds to the depth of the least common ancestor of u and v. Furthermore, the coordinate length $|id(u)|$ gives the depth of u in the tree. Now, the shortest path from u to v consists of the path from u to their least common ancestor w and the path from w to v in the tree. The length of this path is given by the distance function

$$\delta_X(id(u), id(v)) = |id(u)| + |id(v)| - 2cpl(id(u), id(v)). \tag{7.5}$$

Therefore, Equation 7.5 defines a distance such that the PIE embedding is indeed greedy. In addition to being greedy, the embedding is also isometric to the shortest path metric in the tree, i.e., the length of the shortest path between two nodes in the tree is equal to the distance of the nodes' coordinates. Furthermore, the disposal of weights reduces the encoding complexity of the coordinates. For a tree of depth d_{max} with a maximal number of children c_{max}, the length of coordinates is bound by $\mathcal{O}\left(d_{max} \log c_{max}\right)$.

With the assumption of a logarithmic depth, our modified scheme reduces the encoding complexity from $\mathcal{O}\left(\log^3 n\right)$ to $\mathcal{O}\left(\log^2 n\right)$ bits for a topology of n nodes.

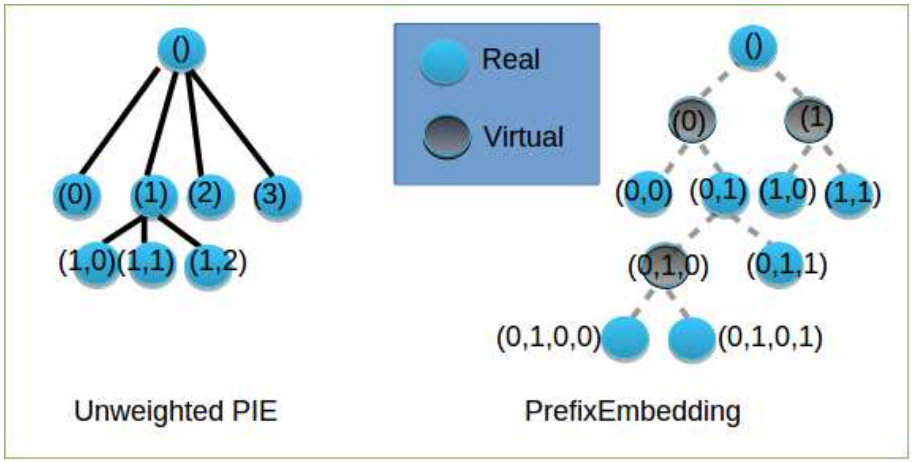


Figure 7.5: Greedy embeddings: PIE embedding modified for unweighted graphs (left), and Prefix Embedding, a variant of PIE allowing for content addressing (right); In PrefixEmbedding, each physical node is represented by several virtual nodes in order to assign binary addresses, i.e., a node with c children in the PIE embedding on the left is replaced by virtual nodes forming a maximally binary tree of depth $\lceil \log c \rceil$. For example, the root node on the left corresponds to a binary tree with node addresses $(), (0)$ and (1) due to its 4 children, who are then assigned the coordinates on the second level of the virtual tree (adapted from [82]).

7.4.2 PrefixEmbedding

While the PIE embedding offers a possibility to efficiently discover short paths within graphs by assigning coordinates and routing based on these coordinates, assigning content to nodes based on these coordinates is not straight-forward. In particular, the coordinates produced by the PIE embedding are integer-valued vectors, whereas keys for content addressing are usually binary hashes. As presented in [82], we could construct keys in the form of integer-valued vectors from binary hashes by representing each k bits as an integer. Based on these keys, we would then map content to nodes by determining the closest node to the key according to Equation 7.5. However, as the number of children and hence the size of the integers varies, there is no universally sensible choice for k. Höfer et al. solve this issue by representing the spanning tree as a virtual binary tree. Then the keys can be mapped in their unaltered form [82]. Assuming that the length of the keys exceeds the depth of the binary tree, a key is mapped to the node whose coordinate is the longest common prefix to the key, thus motivating the name *PrefixEmbedding*.

We now give a more precise account of the above idea. The idea of the approach is to replace each internal node u, i.e., a node with $c > 0$ children, by a set of virtual nodes forming a binary tree. The binary tree is a maximally balanced binary tree, i.e. any tree such that the level of two leaves differ by at most 1, with u as the root and the children as leaves. So, u constructs such a tree and then maps each child to one leaf. The coordinate of the child is then derived as $id(u)$ concatenated with a bit sequence corresponding to the position of the leaf in the binary tree. So, if a node has c children, the first $2c - 2^{\lceil \log_2 c \rceil}$ children receive an address with $\lceil \log_2 c \rceil$ additional bits, the others receive $\lceil \log_2 c \rceil - 1$ additional bits. The construction ensures that no child coordinate is a prefix of any other child coordinate.

The pseudocode of the embedding is displayed in Algorithm 7.1. Initially, the root coordinate is assigned and the root is added to a queue for further processing (Lines 3-4). Then, in each iteration, a node u is removed from the queue (Line 6). If u has children, their coordinates are computed by constructing the binary tree (Line 8), mapping the children to the leaves of the binary tree (Line 9), and assigning each child's coordinate as the concatenation of the parent coordinate and the encoding of the corresponding leaf node in the binary tree (Line 11). Afterwards, the children are added to the queue for further processing. The algorithm terminates after all coordinates have been assigned. For illustration, we give an example in Figure 7.5.

The introduction of virtual nodes has various effects on the distance of nodes and consequently the routing. First, PrefixEmbedding has the disadvantage of losing the isometric property of the PIE embedding as several virtual nodes are represented by the same physical node. The distances given by the embedding are upper bounds on the length of the path between a source-destination pair in the tree. For example, in Figure 7.5, the number of physical overlay hops between the nodes with coordinates

Algorithm 7.1 computePrefixEmbedding()

```
1: {Given: Graph G=(V,E) with spanning tree T}
2: {children(v): children of node v in T, ||: concatenation}
3: Assign ID_R(r) = ()
4: Queue q = {r}
5: while q is not empty do
6:    u = remove head of q
7:    if |children(u)| > 0 then
8:       Create balanced binary tree B of size |children(u)|
9:       Create mapping map: children(u) → leaves(B)
10:      for v ∈ children(u) do
11:         id(v) = id(u)||map(v)
12:         add v to q
13:      end for
14:   end if
15: end while
```

$(0, 1, 0)$ and $(1, 0)$ is 3 but their distance according to Equation 7.5 is 5, corresponding to the number of hops in the virtual tree.

Furthermore, we have to decide which coordinates are used for routing. If all coordinates are provided to children, topology information in form of the number of children is revealed. If only the root coordinate of the virtual tree is provided, the embedding is not strictly greedy. More precisely, the coordinates of two nodes sharing the same parent can be as close or closer to each other than to the parent coordinate. For instance, on the right of Figure 7.5, the node with coordinate $(1, 0)$ is as close to $(1, 1)$ as its parent with coordinate $()$. As a result, the standard greedy routing protocol fails in the last steps as intermediate internal nodes in the virtual tree do not physically exist.

Nevertheless, we prefer providing only the root coordinate and overcoming the lack of greediness by a slight modification of the routing algorithm. In the modified routing algorithm, a node forwards a request to its parent if both of following conditions are met:

1. a local minima of the distance to the destination is reached, and
2. the current node's coordinate is not a prefix of the destination, indicating that the destination is not its responsibility.

This slight modification guarantees the delivery of the message to the responsible node. Algorithm 7.2 presents the pseudocode of the modified routing algorithm. As for the standard greedy algorithm, the current node u first determines the neighbor *next* closest to the requested key (Line 4). If *next* is closer than u, the message is forwarded (Line 6). Otherwise, u is either responsible for the key (Line 9) or forwards it to a parent (Line 11), because a sibling is responsible. In Section 7.4.4, we prove that the successful message delivery is indeed guaranteed.

Algorithm 7.2 nextHop(BitSequence key, Node u)

```
1: {Given: Graph G=(V,E), assignments id, spanning tree T}
2: {parent(v): parent of node v in T}
3: { N(v): neighbors of node v in G}
4: next = argmin{δ_X(id(v), key) : v ∈ N(u)}
5: if δ_X(id(u), key) > δ_X(id(next), key) then
6:    forward to next
7: else
8:    if id(u) is a prefix of key then
9:       routing terminated
10:   else
11:      return parent(u)
12:   end if
13: end if
```

7.4.3 PrefixSEmbedding

When applying *PrefixEmbedding*, as described in Algorithm 7.1, content keys are mapped to nodes with less than 2 children in the spanning tree. In order to see why internal nodes with a higher number

of children are not assigned any keys, recall that we assume the key length to exceed the length of all coordinates. So, if $id(u)$ is a prefix to a key *key* and u has at least two children, there exists a child node v with a longer common prefix with *key*, as the leaves of the binary tree cover all possible additional bits. Hence, either v or one of v's descendants is responsible for *key*. We resolve this restriction of content storage to leaves by creating an additional virtual node for each internal node. More precisely, each internal node u adds a virtual node without children to its set of children. The keys mapped to this virtual node are then stored by u. We denote the modified algorithm by *PrefixSEmbedding*, due to the use of specific storage coordinates id_S.

In the following, we specify our realization of this dual nature of the coordinates. In order to enable a node u to participate both as a leaf storing content and as an internal node forwarding requests, nodes are assigned two coordinates: a *routing coordinate* $id_R(u)$ and a *storage coordinate* $id_S(u)$. The embedding algorithm now proceeds similarly to Algorithm 7.1 with a few key exceptions. As in the original algorithm, the root r node's routing coordinate is $id_R(r) = ()$. When enumerating its children, an internal node u adds a virtual child u'. Routing coordinates are assigned to u's real children by concatenating $id_R(u)$ and the bit sequence of the corresponding leaf in the maximally balanced binary tree as described above. The coordinate of the virtual node u' is u's storage coordinate, i.e., $id_S(u) = id_R(u')$. If a node v is a leaf, the two coordinates are identical. The pseudocode of *PrefixSEmbedding* is given in Algorithm 7.3. The key differences to *PrefixEmbedding* are the size of the binary tree (Line 8) and the assignment of the additional storage coordinate (Line 14).

Algorithm 7.3 computePrefixSEmbedding()

```
1:  {Given: Graph G=(V,E) with spanning tree T}
2:  {children(v): children of node v in T, ||: concatenation}
3:  Assign id_R(r) = ()
4:  Queue q = {r}
5:  while q is not empty do
6:     u = remove head of q
7:     if |children(u)| > 0 then
8:        Create balanced binary tree B of size |children(u)| + 1
9:        Create mapping map: children(u) ∪ {u} → leaves(B)
10:       for v ∈ children(u) do
11:          id_R(v) = id_R(u)||map(v)
12:          add v to q
13:       end for
14:       id_S(u) = id_R(u)||map(u)
15:    else
16:       id_S(u) = id_R(u)
17:    end if
18: end while
```

The routing algorithm distinguishes between the two coordinates by using the routing coordinate for forwarding a message and the storage coordinate for locally determining if a node is responsible for a key. The pseudocode, presented in Algorithm 7.4, differs from Algorithm 7.1 as that the routing always terminates if the storage coordinate of a node u is a prefix to the key (Line 5). Otherwise, if u is not responsible for *key*, the message is forwarded to the closest neighbor or the parent, as in Algorithm 7.1. Note that nodes are not required to reveal their storage coordinates as they are only used locally, thus preventing neighbors from learning an exact characterization of the keys stored at a node.

Algorithm 7.4 nextHopS(BitSequence key, Node u)

```
1:  {Given: Graph G=(V,E), assignments id_R, id_S, spanning tree T}
2:  {parent(v): parent of node v in T}
3:  { N(v): neighbors of node v in G}
4:  if id_S(u) is a prefix of key then
5:     routing terminated
6:  end if
7:  next = argmin{δ_X(id_R(v), key) : v ∈ N(u)}
8:  if δ_X(id_R(u), key) > δ_X(id_R(next), key) then
9:     forward to next
10: else
11:    return parent(u)
12: end if
```

7.4.4 Analysis

In this section, we show that indeed the modified greedy routing protocol terminates successfully. Furthermore, we relate the expected fraction of content assigned to each node to its coordinate. Based on the results, we formulate expectations for our evaluation in Section 7.5.

Throughout this section, we denote the least common ancestor of s and e by $lca(s, e)$, in accordance with the terminology in [80]. Note that we always define the least common ancestor to be a physical node rather than a virtual node. For example in Figure 7.5, the least common ancestor of the nodes with coordinates $(0, 1, 0, 0)$ and $(0, 1, 0, 1)$ is $(0, 1)$.

Theorem 7.6. *Let* id *be an embedding resulting from Algorithm 7.1 or Algorithm 7.3. Then, Algorithm 7.2 or Algorithm 7.4, respectively, ensure that requests for a key* key *are successfully delivered to the closest node to* key*.*

Proof. Let s denote the source node and the e the node responsible for the key *key*. We first assume that no shortcuts are taken during the routing. We divide the path from s to e in the spanning tree into paths from s to the least common ancestor, denoted *lca*, and from *lca* to e. The path from s up the tree to *lca* can again be split into the path from s to a child c of *lca* and the one-hop path from c to *lca*. Greedy routing from s to c succeeds as each parent node shares the same common prefix length with *key* as s but has less coordinates, thus achieving a lower distance according to Equation 7.5. For the one hop from c to the *lca*, greedy routing might fail, as c might have a longer common prefix with e than its parent *lca*. This failure is handled by forwarding to the parent. On the path from *lca* to e, greedy routing is applied and succeeds as the length of the common prefix increases in each step.

Now, assume that shortcuts are used. Because Algorithms 7.2 and 7.4 only allow for shortcuts if the distance to *key* is decreased and distances are integer-valued, the number of shortcuts is finite. Let s' be the last node reached via a shortcut. Then the request is successfully routed from s' to e by the above argument replacing s by s'. □

The fraction of keys that are mapped to a node u determine the amount of content u stores, assuming a close to uniform mapping from content to keys as provided by e.g., cryptographic hash functions. We show that the expected fraction of content of a node decreases exponentially with the length of its coordinate, thus entailing that the majority of content is stored on leaves close to the root.

Theorem 7.7. *Let id be an embedding resulting from Algorithm 7.1 The fraction $\mu_{Pre}(\mathbb{B}(u))$ of b-bit keys mapped to a node u with $|id(u)| < b$ is*

$$\mu_{Pre}(\mathbb{B}(u)) = \begin{cases} 0, & |children(u)| > 1 \\ \frac{1}{2^{|id(u)|+1}}, & |children(u)| = 1 \\ \frac{1}{2^{|id(u)|}}, & |children(u)| = 0 \end{cases}. \tag{7.6}$$

Proof. Note that u is always responsible for the keys closest to its coordinate $id(u)$ according to the distance in Equation 7.5. The first case follows as all keys are mapped to children of u that have a longer common prefix with the key. If u does not have children, all keys with $id(u)$ as a prefix are assigned to u, which corresponds to the number $2^{b-|id(u)|}$ of possible postfixes for $|id(u)|$ fixed bits. The result follows because

$$\frac{2^{b-|id(u)|}}{2^b} = \frac{1}{2^{|id(u)|}}.$$

Similarly, if u has exactly one child, half of these keys, i.e., those with a 0 after the first $|id(u)|$ bits, are closer to the child's coordinate than to $id(u)$. Hence, the fraction of keys mapped to u is reduced by a factor 2 in contrast to a leaf node u. □

Theorem 7.8. *Let id be an embedding resulting from Algorithm 7.3. The fraction $\mu_S(\mathbb{B}(u))$ of b-bit keys mapped to a node u with $|id_S(u)| < b$ is*

$$\mu_S(\mathbb{B}(u)) = \frac{1}{2^{|id_S(u)|}}. \tag{7.7}$$

Proof. The result follows analogously to the case $|children(u)| = 0$ in Theorem 7.7, because the coordinate $id_S(u)$ is assigned to a leaf of the virtual tree. □

By Equation 7.6, *PrefixEmbedding* assigns the highest fraction of keys to the leaf node closest to the root in the virtual binary tree. In contrast, Equation 7.7 shows that *PrefixSEmbedding* assigns the highest fraction of keys to either the root node or any of its children that is a leaf, because the virtual root node is the leaf closest to the root unless one of the real children is also a leaf. As a consequence, while *PrefixSEmbedding* includes all nodes in the key assignment rather than only a fraction, it is still likely to exhibit a higher maximum with regard to the fraction of assigned keys.

7.5 Simulation

In this section, we exemplary present the results of applying the previously introduced content addressing schemes to our three considered topologies. Given the theoretical results in Section 7.3, we expect that the content addressing is highly unbalanced and thus individual nodes are responsible for the majority of content, entailing overload and congestion.

7.5.1 Simulation Model and Set-up

We evaluated the content addressing using a snapshot-based analysis as introduced in Section 5.2.1. For this purpose, we executed the following three steps:

1. Construct a spanning tree for a given topology by selecting a random root and using a breadth-first search to establish the tree,
2. Generate the embedding id by Algorithm 7.1 (*PrefixEmbedding*) as well as the embeddings id_R and id_S by Algorithm 7.3 (*PrefixSEmbedding*), and
3. Compute the fraction of keys nodes are responsible for as by Equation 7.6 and Equation 7.7, respectively.

During the tree construction, nodes choose their parent randomly from those neighbors with the shortest path to the root. For Algorithm 7.3, the virtual binary tree is constructed such that the leaf corresponding to the current node is at a maximal depth, thus slightly reducing the fraction of keys mapped to the corresponding leaf.

Our evaluation utilized the three social network topologies introduced in Section 4.2, namely the Facebook graph *FB*, the special-purpose social network *SPI*, and the Web-of-Trust *WOT*. Our results were averaged over 20 runs. For each run, we obtained the fraction of keys per node, sorted the resulting list in decreasing order of the fraction of keys, and then averaged over the sorted list.

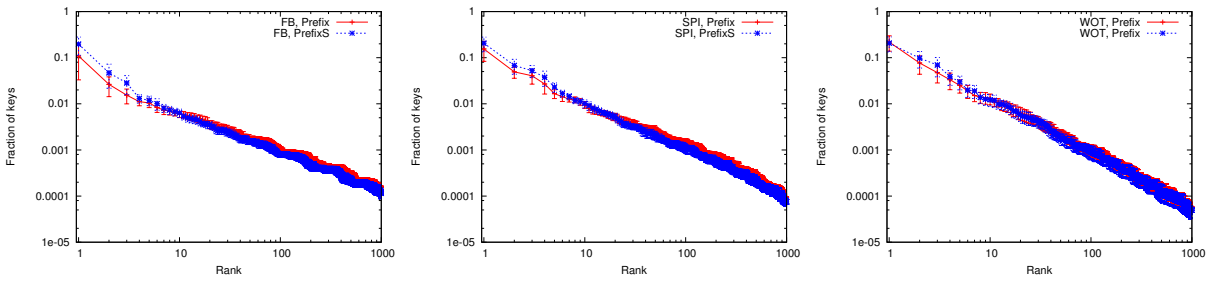
7.5.2 Expectations

We expected the content addressing to be highly unbalanced, because spanning trees on social networks are inherently unbalanced and the fraction of keys is directly related to the depth of a node in the tree. In particular, we expected the maximal fraction to be especially high for *PrefixSEmbedding*, as indicated by Theorem 7.8. Note that the storage coordinate of a root with c children is of length $l = \lceil \log(c+1) \rceil$, i.e., the depth of the binary tree created for the coordinate assignment of the root node. Hence, by Equation 7.7, the maximal fraction of keys is at least 2^{-l}. For scale-free graphs, the average degree is independent of the number of nodes, so that *PrefixSEmbedding* is expected to result in a great imbalance with regard to the load assigned to nodes. In contrast, the maximal fraction of keys for *PrefixEmbedding* depends on the depth of the first node with less than 2 children. However, as all our exemplary graphs exhibit a high number of nodes with only one neighbor, the first such node was expected to be close to the root. So, we expected a only slightly lower maximal fraction of keys for *PrefixEmbedding* than for *PrefixSEmbedding*.

In addition to those general expectations, we evaluated the differences between the three considered topologies. In particular, *WOT* exhibits a high fraction of low degree nodes, as displayed in Table 4.2 and Figure 4.2 in Section 4.2. More than half of the nodes have at most 4 neighbors, *PrefixSEmbedding* maps at least 1/8 of the keys to the root in the light of the above argument for $c = 4$. As a consequence, the maximal fraction of keys stored at a node exceeds 1/16 in a graph of more than $40,000$ nodes. We expected that the two online social networks *FB* and *SPI* exhibited slightly less drastic results due to their less skewed degree distributions. Nevertheless, the resulting spanning trees were bound to be unbalanced as well, hence resulting in an unbalanced content addressing. Because of the dependence of the maximal fraction of keys on the degree of the root, we expected a high variance corresponding to the high variance of the degree distribution.

7.5.3 Results

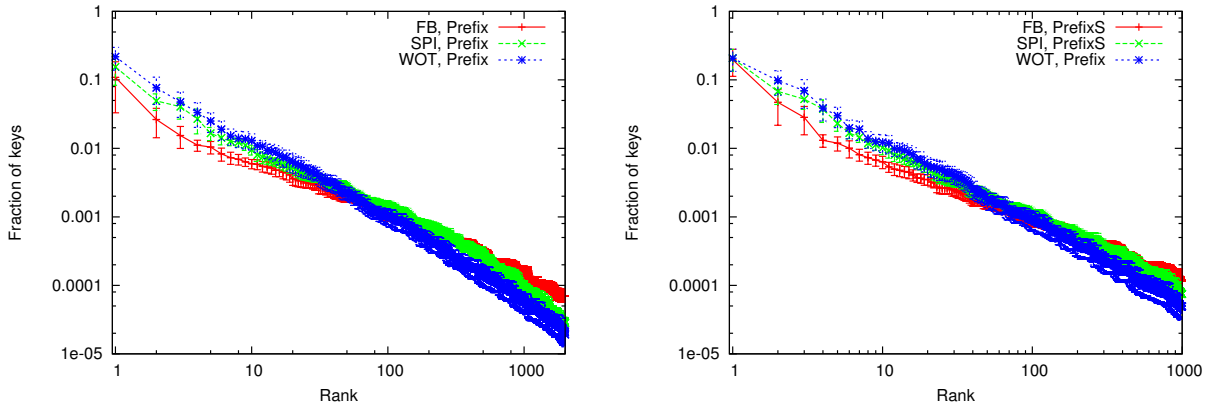
Indeed, the results agreed with our expectations. Figure 7.6 displays the maximal fraction of keys mapped to one node for each of the social graphs with regard to both *PrefixEmbedding* and *PrefixSEmbedding*. For an improved readability, we restrict the displayed distributions to 1000 nodes with the highest fraction of keys. The maximal fraction of keys observed in *PrefixSEmbedding* was always higher than in *PrefixEmbedding* for *FB* (Figure 7.6a) and *SPI* (Figure 7.6b). More concretely, *FB* exhibits maximal fractions of roughly 19% and 10% and *SPI* of 15% and 20% with regard to *PrefixSEmbedding* and *PrefixEmbedding*, respectively. In contrast, the maximal fractions for *WOT* (Figure 7.6c) were nearly identical, with 21.8%



(a) FB (b) SPI (c) WOT

Figure 7.6: Fraction of keys assigned to nodes (restricted to 1,000 nodes with the highest fraction) by *PrefixEmbedding* and *PrefixSEmbedding* for three social graphs

for *PrefixEmbedding* and 20.7% for *PrefixSEmbedding*. The reason for the observed differences lies in the high fraction of nodes with only one neighbor in *WOT*, entailing a high likelihood for leaves on the first level of the tree. Indeed, the inclusion of the root as an additional child node reduces the fraction of keys mapped to those leaves, which explains the slightly lower maximal fraction for *PrefixSEmbedding*. Furthermore, the close to linear decrease observed in the loglog scale indicates that the fraction of keys decreases exponentially with the rank of the node. If indeed the dependence between rank and fraction of keys is exponentially distributed, the maximal fraction of keys is largely independent of the network size. In other words, the maximal fraction scales with $\theta(1)$ rather than the required $\mathcal{O}\left(\frac{\log n}{n}\right)$.



(a) PrefixEmbedding (b) PrefixSEmbedding

Figure 7.7: Comparison of three social graphs with regard to the fraction of keys assigned to nodes (restricted to 1,000 nodes with the highest fraction) for *PrefixEmbedding* and *PrefixSEmbedding*

The differences between the 3 graphs are illustrated in Figure 7.7. Figure 7.7a displays the fraction of keys assigned to the top 1000 nodes for *PrefixEmbedding*, whereas Figure 7.7b displays the results for *PrefixSEmbedding*. As expected, the fraction of the keys in *PrefixEmbedding* was closely related to the degree distribution, so that the distribution of keys was most balanced in *FB*, followed by *SPI*, and last *WOT*. The topology of the social network also affected *PrefixSEmbedding*, albeit in a less drastic manner. *PrefixSEmbedding* mapped roughly 20% of all keys to one node for all topologies, but the fractions assigned to less overloaded nodes varied between the different topologies. The general high variance of the assigned fractions, as indicated by the errorbars for the 95% confidence intervals, was due to the high differences with regard to the degree of the root.

In summary, the simulation complements our theoretical results by illustrating the inability of tree-based greedy embeddings to provide balanced content addressing.

7.6 Discussion

We have seen that tree-based greedy embeddings do not provide balanced content addressing and are inherently unable to do so without relaxing our remaining requirements. In particular, our theoretical analysis formally ascertains that balanced content addressing can only be guaranteed if nodes reveal

an estimate of their number of descendants. Besides the potential leakage of private information, the algorithm's dependency on the number of descendants allows a malicious node to assign the majority of content to compromised nodes and thus censor an arbitrary high fraction of published content. A secure consensus protocol might potentially mitigate this attack but such protocols are very costly, increasing the average stabilization complexity to be at least linear in the number of nodes, contradicting our requirements. Apart from this provable weakness, it remains unclear how reliably distributed consensus protocols can be implemented in F2F overlays. In summary, we have shown the impossibility of guaranteeing balanced content addressing without introducing either severe vulnerabilities or exhaustive stabilization complexity.

However, tree-based greedy embeddings can offer an efficient routing protocol. Complementary, virtual overlays provide balanced content addressing but are inefficient due to their maintenance-intensive tunnels. In other words, they maintain explicitly defined routes between pairs of nodes rather than applying an additional coordinate-based routing protocol. Greedy embeddings present a promising candidate for such a routing protocol. It remains to be seen if they can be modified such that they achieve robustness, censorship-resilience, anonymity, and membership-concealment. We show how to realize these requirements in the next chapter.

Chapter 8

VOUTE

Up to now, we have evaluated the existing approaches and identified their principal weaknesses with regard to performance and attack resistance. In particular, we have shown that greedy embeddings fail to fulfill our requirements despite their high efficiency. Now, we present VOUTE: Virtual Overlays Utilizing Tree Embeddings, a novel F2F overlay design that satisfies our requirements. In a nutshell, VOUTE constructs multiple greedy embeddings allowing for fast communication between arbitrary pairs of nodes. Content addressing is achieved by a virtual overlay leveraging the greedy routing of the embedding for communication between virtual neighbors. The distribution of the traffic on multiple embeddings increases the robustness and censorship-resistance. Anonymity is realized by generating anonymous return addresses, which resume the role of coordinates during routing and prevent a local adversary from identifying the communicating parties.

In the following sections, we present the details of our design and its evaluation. In Section 8.1, we start by giving an overview of our design. We detail our design in Section 8.2, starting with the tree construction and embedding, followed by the return address generation, and the virtual overlay. During the subsequent evaluation, we consider efficiency and scalability in Section 8.3, robustness and censorship-resistance in Section 8.4, and anonymity and membership-concealment in Section 8.5. We conclude that VOUTE fulfills our requirements and discuss future work in Section 8.6. The results are accepted for publication in INFOCOM 2016 [130].

8.1 Idea

In the following, we give an overview of how VOUTE fulfills the requirements from Section 2.6, addressing stabilization, robustness and censorship-resistance, anonymization and membership-concealment, and balanced content addressing. An overview of the different layers of our system, the F2F overlay, the embeddings, and the virtual overlay, is displayed in Figure 8.1.

Stabilization: The stabilization of embedding-based overlays has not been considered in detail because the related work considers Internet routing or static wireless sensor networks as applications [87, 139]. In these applications, node and edge failures or additions are few. Hence, recomputing the complete embedding whenever a topology change occurs is acceptable. In contrast, F2F overlays change frequently, so that rebuilding the complete spanning tree whenever the topology changes results in a high communication complexity. In practice, complete reconstructions are even expected to be unfeasible due to the fact that the interval between two topology changes is bound to be lower than the construction duration in large overlays.

Hence, our approach utilizes an embedding such that only nodes disconnected from the tree have to select a new parent. We show that the stabilization complexity of the trees is indeed polylog as required. Our simulation study on real-world social networks ascertains that the stabilization overhead is usually low, similar to the overhead of discovering one route.

Robustness and Censorship-Resilience: Achieving robustness and censorship-resistance in tree-based embeddings is challenging. Adversaries can abuse the structure of the trees to obtain an important position and then drop all messages. Hence, in one tree-based embedding, requests can only be routed successfully if either source and destination are in the same subtree or the routing algorithm discovers a shortcut between the subtree of the source and the destination. We aim to increase the probability that the routing terminates successfully.

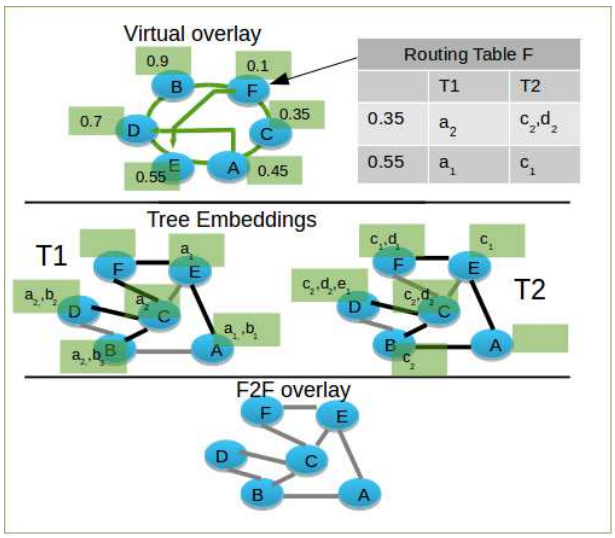


Figure 8.1: Layers of VOUTE: 1) F2F overlay as restricted topology, 2) Tree embeddings $T1$ and $T2$ offer addressing for messaging, 3) Virtual overlays leveraging the embeddings' routing algorithm offer content sharing

First, we construct multiple embeddings in parallel, similar to the parallel realities in CAN [124] During construction, the spanning trees are built in such a manner that nodes prefer different parents in all embeddings. In this manner, both the probability that two nodes are in the same subtree in at least one embedding and the probability to find shortcuts in at least one embedding are increased.

Second, the current routing algorithm fails as soon as a local optimum with regard to the distance is reached. However, it might be possible to use an alternative route offered by one of the previous nodes on the route that has several neighbors closer to the destination. We hence increase the probability of successful routing by adding a backtracking phase to the routing to search for alternative routes.

Third, the tree distance prefers nodes close to the root, resulting in a high importance of nodes close to the root. If an attacker is in such a position, it can intercept a disproportional high fraction of requests. We modify the distance function such that nodes in the same subtree as the destination are always contacted first, even if forwarding via the root would result in a shorter route. So, we reduce the efficiency in favor of reducing the impact of obtaining strategically important positions.

In summary, we improve the resilience in a variety of ways. The price of the improvement is the increased communication complexity. Nevertheless, routing and stabilization complexity still scale logarithmically in the number of nodes and at most quadratic in the number of trees. We evaluate robustness and censorship-resistance in an extensive simulation study, indicating that both are improved in comparison to related approaches. The system can withstand large-scale failures and powerful attacks, reliably providing success ratios of more than 95%.

Anonymity and Membership-Concealment: The problem of achieving receiver anonymity in a network embedding is both particularly important and challenging. In an embedding, coordinates are used to address nodes and hence users. Though only direct neighbors can directly map the embedding coordinate to a real-world identity, arbitrary participants can reconstruct the social graph based on the revealed coordinates. Participants might then be identified from the graph structure [115]. Hence, the use of coordinates as addresses in requests prevents both receiver anonymity and membership-concealment. Nevertheless, efficient communication in an embedding requires the use of addresses.

We solve the apparent contradiction between efficiency and anonymity by designing a protocol to transform coordinates to anonymous return addresses and modifying the routing protocol to work on these return addresses. We show that the return addresses provide possible innocence against a local adversary. Furthermore, they obfuscate the structure of the social graph without increasing the communication complexity.

Balanced Content Addressing: Last, network embeddings allow communication between nodes in the F2F overlay but inherently fail to provide balanced content addressing, as shown in Chapter 7. For this reason, we construct a virtual overlay on the basis of greedy embeddings. More exactly, virtual neighbors communicate by leveraging the routing algorithm for the embedding.

We show that the routing complexity for content discovery scales polylog as required. Our simulation

study indicates that the routing complexity is only minimally increased in comparison to the tunnel-based approach, whereas the stabilization complexity is greatly reduced.

Combined, the above protocols result in a novel F2F overlay that is i) efficient and scalable, ii) robust and censorship-resistant, and iii) anonymous and membership-concealing.

8.2 Design

Our main contribution lies in proposing multiple greedy embeddings with anonymous return addresses and a virtual overlay on top of the embeddings. In the following, we present our system, in particular

- a spanning tree construction and stabilization algorithm for multiple parallel trees enabling robustness and censorship-resistance,
- an embedding algorithm enabling scalable messaging as well as improved censorship-resistance through modified distance functions and routing algorithms,
- an address generation algorithm $\mathbf{AdGen}_{node}$ enabling receiver anonymity and membership-concealment, and
- a virtual overlay design based on embeddings enabling balanced content addressing, efficient content retrieval, and efficient stabilization.

Throughout this section, let b be a sufficiently large integer, $PRNG$ a pseudo-random number generator with values in $\mathbb{Z}_{2^b}$, and $h : \{0,1\}^* \to H$ a cryptographic hash function.

8.2.1 Tree Construction and Stabilization

In this section, we show how we construct and stabilize γ parallel spanning trees. In the next section, we then describe our coordinate assignment on the basis of these trees. We aim to increase the robustness and censorship-resistance by using multiple trees. In order to ensure that the trees indeed offer different routes, our algorithm encourages nodes to select different parents in each tree if possible. Our algorithm design follows similar principles as the provable optimally robust and resilient tree construction algorithm for P2P-based video streaming presented in [37]. However, the algorithm assumes that nodes can change their neighbors. Thus, we cannot directly apply the algorithm nor the results. In the following, we first discuss the tree construction and then the stabilization.

Tree Construction: We divide the construction of a tree into two phases: i) selecting the root, and ii) building the tree starting from the root. We reviewed possibilities for the root election in Section 2.2.2. We can apply one of them, e.g., [120], which achieves a communication complexity of $\mathcal{O}(n \log n)$. Our own contribution lies in the tree construction after the root has been chosen.

We now shortly describe the idea of our algorithm before detailing the actual algorithm. A node u that is not the root receives messages from its neighbors when they join a tree and become potential parent nodes. There are two questions to consider when designing an algorithm governing u's reaction to such messages, called invitations in the following. First, u has to decide if and when it accepts an invitation. Second, u has to select an invitation in the presence of multiple invitations.

For the second question, u always prefers invitations from nodes that have been their parent in less trees with the goal of constructing different trees and increasing the overall number of possible routes. Increasing the number of routes allows the use of alternative routes if the request cannot be routed along the preferred route due to a failed or malicious node. If two neighbors are parents in the same number of trees, u can either select one randomly or prefer the parent closer to the root. Choosing a random parent reduces the impact of nodes close to the root but is likely to lead to longer routes and thus a lower efficiency.

Coming back to the first question of if and when u accepts invitations, u always accept an invitation of a neighbor v that is not yet a parent of u in any tree in order choose different parents as often as possible. In contrast, if v is already a parent, u might wait for the invitation of a different neighbor. However, it is unclear if it is possible for all neighbors of u to ever become a parent. For example, a neighbor of degree 1 is only a parent if it is the root. In order to overcome this dilemma, u periodically probabilistically decides if it should accept v's invitation or wait for another invitation. So, u eventually accepts an invitation but does provide alternative parents the chance to send an invitation.

Now, we describe the exact steps of our round-based algorithm for the construction of γ parallel spanning trees. The pseudocode governing the behavior of one node u in a round is given in Algorithm

8.1. After a node u is included in the i-th tree, u sends invitations (i, u) to all its neighbors inviting them to be its children in tree i. When u receives an invitation (j, w) for the j-th tree from a neighbor w, it saves the invitation if it is not yet contained in tree j and otherwise stores it. The invitation can still be used if u has to modify its parent selection later. In each round, a node u considers all invitations for trees it is not yet part of. Let $pc(v)$ denote number of the trees T such that the neighboring node v is a parent of u in T. If u has received invitations from neighbors v with minimal $pc(v)$ among all neighbors, u accepts one of those invitations (Lines 1-3). In the presence of multiple invitations, we experiment with two selection strategies: i) Choosing a random invitation, and ii) Choosing a random invitation from a node on the lowest level. The latter selection scheme requires that the invitations also contain the level of the potential parent node in the tree. If u does not have an invitation from any node with minimal $pc(v)$, u nevertheless accepts an invitation with probability q in order to guarantee the termination of the tree construction. If u accepts a parent, it selects a node v that has offered an invitation and has the lowest $pc(v)$ among neighbors with outstanding invitations (Lines 7-8). In this manner, we guarantee the convergence of tree construction.

The acceptance probability q is essential for the diversity and the structure of the trees: For a high q, nodes quickly accept invitations leading to trees of a low depth and thus short routes. However, in the presence of an attacker acting as the root of all or most trees, the trees are probably close to identical, resulting in a low censorship-resistance. A lower acceptance probability q increases the diversity but entails longer routes. Thus, a low q results in a higher communication complexity and at some point decreases the robustness due to the increased likelihood of encountering failed nodes on a longer route. In Section 8.3.1, we show that the constructed trees are of a logarithmic depth such that we indeed maintain a routing complexity of $\mathcal{O}(\log n)$.

Note that Algorithm 8.1 does not assume that all trees are constructed at the same time. Rather, individual trees can be (re-)constructed while the remaining trees impact the parent choice in the new tree but remain unchanged.

Algorithm 8.1 constructTreeRound()

{*Internal state: Set I of invitations, acceptance probability q, $pc : N_u \to \mathbb{N}_0$ number of times neighbor is parent, Selection strategy W*}
1: $PP \leftarrow \{(i, w) \in I : \forall \mathbf{v} \in \mathbf{N_u} : pc(w) \leq pc(v)\}$
2: **if** PP is not empty **then**
3: Select invitation in PP to answer according to W
4: **else**
5: $r \leftarrow$ uniform random number
6: **if** $r \leq q$ **then**
7: $PQ \leftarrow \{inv = (i, w) \in I : \forall (\mathbf{j}, \mathbf{v}) \in \mathbf{I} : pc(w) \leq pc(v)\}$
8: Select invitation in PQ to answer according to W
9: **end if**
10: **end if**

Stabilization: Now, we consider the stabilization of the trees when nodes join and leave. Stabilizing the trees efficiently, i.e., repairing them locally rather than reconstructing the complete tree whenever the topology changes, is essential for efficiency. Joining nodes can be integrated in a straight-forward manner by connecting to their neighbors as children, again trying to maximize the diversity of the parents. For this purpose, nodes record the time, i.e., the round in our abstract time model, they joined the tree. Now, when a new node u joins, it requests its neighbors' coordinates and these timestamps for all trees. Based on this information, u can simulate Algorithm 8.1 locally, ensuring that its expected depth in the tree is unaffected by its delayed join. When a node departs, all its children have to choose a different parent and inform their descendants of the change. In order to prevent a complete subtree from being relocated at an increased depth, the descendants may also select a different parent. The selection of the new parent again follows Algorithm 8.1 but only locally re-establishes the trees affected by the node departure.

We show that the stabilization complexity considering any node but the root is linear in the terms of average depth of the node in the trees. The desired upper bound on the expected stabilization complexity follows from showing that this depth remains logarithmic in the network over an extended period of time. Only if the root departs, i.e., with probability $1/n$ assuming a uniform probability in choosing the departing node, the tree has to be reconstructed. Thus, the complexity depends on the root selection protocol, which can have complexity $\mathcal{O}(n \log n)$, as detailed in Section 2.2.2. We formally prove that the above stabilization algorithm indeed introduces only logarithmic complexity in Section 8.3.1.

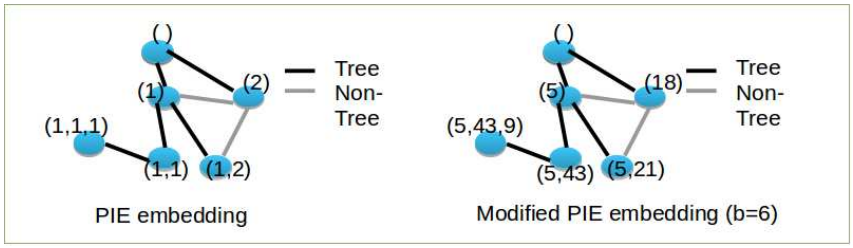


Figure 8.2: Original PIE and modified PIE coordinates using $b = 6$-bit numbers

8.2.2 Embedding and Routing

In this section, we show how to assign coordinates in a spanning tree and how to route based on these coordinates. As we want to prevent an attacker from guessing the coordinate of a receiver, we require a certain degree of in-determinism in the coordinate assignment. We thus choose a slightly modified version of the unweighted PIE embedding [80], which we have introduced in Section 7.4.1. Our main modification lies the use of in-deterministic coordinates in order to prevent an adversary from guessing the coordinate and thus undermining the anonymization schemes presented in the next section. In addition to the tree distance in [80], we also present a second distance preferring nodes with a long common prefix and thus avoiding routes via nodes close to the root whenever possible. In this manner, we increase robustness and censorship-resistance, because the routing algorithm considers alternative routes and the impact of strategically choosing a position close to the root is reduced. Our routing algorithm then uses greedy routing based on any of the two distances and backtracking. In the following, we subsequently present the embedding algorithm, the distance functions, and the routing algorithm with backtracking.

Embedding Algorithm: Our algorithm assigns the coordinates on each of the γ trees independently, so that we only consider one embedding id. The coordinate assignment starts at the root and then spreads successively throughout the tree. After a spanning tree has been established, the root r is assigned an empty vector as a coordinate $id(r) = ()$. In the next step, each child v of the root generates a random b-bit number $a \in \mathbb{Z}_{2^b}$ such that its coordinate is $id(v) = (a)$. Here, our algorithm differs from the PIE embedding because it uses random rather than consecutive numbers, thus preventing an adversary from guessing the coordinate in an efficient manner. Subsequently, nodes in the tree are assigned coordinates by concatenating their parent's coordinate with a random number. So, upon receiving its parent coordinate $id(p(v)) = (a_1, \ldots, a_{l-1})$, a node v on level l of the tree obtains its coordinate $id(v) = (a_1, \ldots, a_{l-1}, a_l)$ by adding a random b-bit number a_l. The coordinate space is hence given by all vectors consisting of b-bit numbers, i.e., $\mathbf{X} = \{(a_1, \ldots, a_{l-1}, a_l) : l \in \mathbb{N}_0, a_i \in \{0,1\}^b\}$. Figure 8.2 displays the differences between the original PIE embedding and our variation.

Note that the independent random choice of the b-bit number $a \in \mathbb{Z}_{2^b}$ might lead to two nodes having the same coordinate. Thus, b should be chosen such that the chance of equal coordinates should be negligible. If two children nevertheless select the same coordinate, the parent node inform one of them to adapt its choice. Note that allowing the parent to influence the coordinate selection in this manner does not really increase the vulnerability to attacks, as the parent can achieve at least the same damage by constantly changing its coordinate. Such constant changes can be detected easily, so that nodes should stop selecting such nodes as parents. In general, by moving the choice of the last coordinate element from the parent to the child, we automatically reduce the impact of a malicious parent as it cannot determine the complete coordinate of the child.

Distances: We still need to define distances between coordinates in order to apply greedy routing. For this purpose, we consider two distances on $\mathbf{X}$. Both rely on the common prefix length $cpl(x_1, x_2)$ of two vectors x_1 and x_2 and the coordinate length $|x_1|$.

First, we consider the tree distance δ_{TD} from [80], which gives the length of the path between the two nodes in the tree, i.e.,

$$\delta_{TD}(x_1, x_2) = |x_1| + |x_2| - 2cpl(x_1, x_2). \tag{8.1}$$

Secondly, the common prefix length can be used as the determining factor in the distance function, i.e., for a constant L exceeding the length of all node coordinates in the overlay, we define

$$\delta_{CPL}(x_1, x_2) = \begin{cases} L - cpl(x_1, x_2) - \frac{1}{|x_1|+|x_2|+1}, & x_1 \neq x_2 \\ 0, & x_1 = x_2 \end{cases}. \tag{8.2}$$

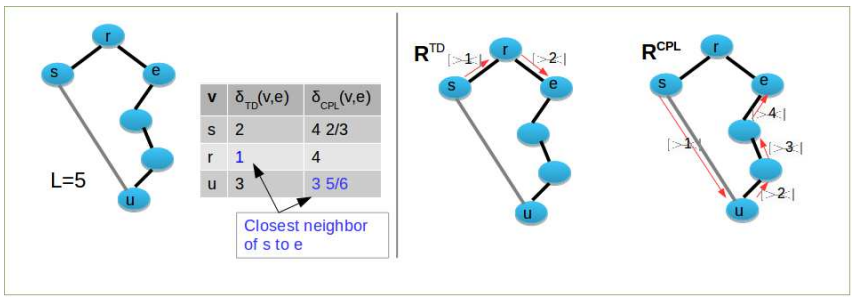


Figure 8.3: Tree distance (TD) δ_{TD} and common prefix length based distance δ_{CPL} when routing from node s to e: δ_{CPL} prefers nodes in the same subtree as the destination, leading to better censorship-resistance at the price of longer routes. The table gives the distances in the first hop for s and its neighbors r and u.

The reason for using the common prefix length rather than the actual tree distance is the latter's preference of routes passing nodes close to the root in the tree. In this manner, nodes on these routes are very influential, so that adversaries close to the root have a large impact. In contrast, δ_{CPL} prefers possibly longer routes by always forwarding to a node within the same subtree as the destination and avoids central nodes in the tree. An example of the difference between the two distances and the impact on the discovered routes is displayed in Figure 8.3.

Algorithm 8.2 route()

{*Input: current node u, message msg from node w, tree index i, target coordinate* x_i}
{*Internal state: set* $S(msg)$ *of nodes u forwarded mess to, predecessor* $pred(msg)$*, distance* δ}
if $id_i(u) == x_i$ **then**
  Routing succeeds
else
{*Store predecessor unless backtracking*}
  if not $S(msg)$ contains w **then**
    $pred(msg) \leftarrow w$
  end if
{*Determine closest neighbors*}
  $C \leftarrow argmin_{v \in N_u \setminus S(msg)} \delta(id_i(v), x_i)$
  $next \leftarrow$ random element in C
  if $\delta(id_i(v), x_i) > \delta(id_i(next), x_i)$ **then**
    Forward msg to $next$ {*Forward if improvement*}
  else
    if $pred(msg)$ is set **then**
      Forward msg to $pred(msg)$ {*Backtrack*}
    else
      Routing failed
    end if
  end if
end if

Greedy Routing in Multiple Embeddings: We route in $1 \leq \tau \leq \gamma$ trees in parallel. More precisely, given a vector of coordinates $(id_1(e), \ldots, id_\gamma(e))$, the sender s selects τ coordinates and sends a request for each of them. s can either select τ embeddings uniformly at random or choose the embeddings so that the distance of the neighbor v_i with the closest coordinate to $id_i(e)$ is minimal. The latter choice might result in shorter routes due to the low distance in the embedding.

The routing processes in each embedding independently. Nodes forward the request to the neighbor with the closest coordinate in the respective embedding. Thus, in order for the nodes on the route to forward the request correctly, the request has to contain both the coordinate $id_i(v)$ and the index i of the embedding. In practice, we can achieve a performance gain by including multiple coordinates and embedding indices in one message if the next hop in two or more embeddings are identical. For now, we assume that one message is sent for each embedding.

We optionally increase the robustness and censorship-resistance of the routing algorithm by allowing backtracking if the routing gets stuck in a local minimum of the distance function due to failures or intentional refusal to forward a request. For this purpose, all nodes remember their predecessor on the

routing path as well as the neighbors they have forwarded the request to. If all neighbors closer to the target have been considered and have been unable to deliver the request, the node reroutes the request to its predecessor for finding an alternative path. The routing is thus only considered to be failed if the request returns to its source s and cannot be forwarded to any other neighbor. In this manner, all *greedy paths*, i.e., all paths with a monotonously decreasing distance to the target, are found.

Algorithm 8.2 gives the pseudo code describing one step of the routing algorithm, including the backtracking procedure. When receiving a message msg, the node u first checks if it is the receiver of msg, thus successfully terminating the routing (Line 2). If u is not the receiver, it determines if the routing is currently in the backtracking phase by checking if u has previously forwarded msg to the sender w. Otherwise, it stores the sender of msg as a predecessor for potential later backtracking (Line 5). In the manner of greedy routing, u selects the closest neighbor to the target coordinate. In the presence of several closest neighbors, u picks one of them uniformly at random (Lines 7-8). Note that in the presence of failures, the embedding can lose its greediness. Hence, to avoid loops, u only forwards the request to the selected neighbor if it is indeed closer to the destination than u (Line 10). Otherwise, u contacts its predecessor (Line 13) or forfeits the routing if no such predecessor exists (Line 15), i.e., if u is the source of the request.

This completes the description of the routing and stabilization functionalities. However, up to now, we used identifying coordinates rather than anonymous addresses.

8.2.3 Anonymous Return Addresses

In this section, we introduce our address generation algorithm $\mathbf{AdGen}_{node}$ for generating anonymous return addresses, which do not reveal the receiver of the request. For this reason, we call the generated addresses *route preserving (RP) return addresses*. Based on these return addresses, we specify two routing algorithms $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ for routing a request based on an included return address. These return addresses allow a node to determine the common prefix length of their neighbor's coordinates and the receiver coordinate, which allows the node to determine the closest neighbor. Hence, $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ correspond to Algorithm 8.2 for the two distance function δ_{TD} and δ_{CPL} when using return addresses rather than receiver coordinates. After describing the algorithm, we show that the return addresses indeed preserve routes.

Return Address Generation: Return addresses are generated in three steps:

1. Padding the coordinate
2. Applying a hash cascade to obtain the return address
3. Adding a MAC

Algorithm 8.3 displays the pseudo code of the above steps.

Algorithm 8.3 generateRP()

```
   {Input: coordinate x = (a_1, ..., a_l), seed s, s_pad}
   {Internal State: key K_MAC(v), h, PRNG}
 1: k̃ ← PRNG(s)
 2: d_1 ← h(k̃ ⊕ a_1)
 3: for j = 2 ... L do
 4:    if j ≤ l then
 5:       a'_j ← a_j
 6:    else
 7:       a'_j ← PNRG(s_pad + j) {Padding}
 8:    end if
 9:    d_j ← h(d_{j-1} ⊕ a'_j) {Hash cascade}
10: end for
11: mac ← h(K_MAC(v)||d_1||d_2|| ... ||d_L) {MAC}
12: Publish y = (d_1, ..., d_L), k̃, mac
```

The first step of the return address generation prevents an adversary from identifying coordinates based on their length. A node v pads its coordinate $x = (a_1, \ldots, a_l)$ by adding random elements $a'_{l+1}, \ldots, a'_L$. More precisely, v selects a seed s_{pad} for the pseudo-random number generator $PRNG$

and obtains the padded coordinate $x' = (a'_1, \ldots, a'_l, a'_{l+1}, \ldots, a'_L)$ with

$$a'_j = \begin{cases} a_j, & j \leq l \\ PRNG(s_{pad} \oplus j), & j > l \end{cases}.$$

In order to ensure that the closest node coordinate to x' is indeed x, v recomputes the padding with a different seed if a'_{l+1} is equal to the $l+1$-th element of a child's coordinate [1]. Afterwards, v chooses a different seed s for the construction of the actual return address and generates $\tilde{k} = PRNG(s) \in \tilde{\mathbb{K}} = \mathbb{Z}_{2^b}$. v then executes the local function

$$\begin{aligned} hc : \mathbf{X} \times \tilde{\mathbb{K}} &\to \mathbf{Y} = H^L \\ (x, \tilde{k}) &\mapsto y = (d_1, \ldots, d_L) \\ d_j &= \begin{cases} h(\tilde{k} \oplus a'_1), & j = 1 \\ h(d_{j-1} \oplus a'_j), & j = 2 \ldots L \end{cases}. \end{aligned} \tag{8.3}$$

We call the pair $(y, \tilde{k})$ a *return address*, which can be used to find a route to the node with coordinate x. Before publishing the return address, v adds a MAC $mac(y_i, \mathbb{K}_{MAC}(v)) = h(d_1 || \ldots d_L || \mathbb{K}_{MAC}(v))$ for a private key $\mathbb{K}_{MAC}(v)$ to prevent malicious nodes from faking return addresses and gaining information from potential replies. Last, v publishes the return address $(y, \tilde{k})$ and the MAC.

Routing Algorithms: Now, we determine diversity measures $\delta_{RP-TD} : \mathbf{X} \times \mathbf{Y} \to \mathbb{R}_+$ and $\delta_{RP-CPL} : \mathbf{X} \times \mathbf{Y} \to \mathbb{R}_+$ in order to compare a coordinate c and a return address $(y, \tilde{k})$ with regard to δ_{TD} and δ_{CPL}. The diversity measure then assumes the role of the distance δ in Algorithm 8.2. [2]

In order to define the two diversity measures, note that for any coordinate c and return address $(y, \tilde{k})$ corresponding to the coordinate x, we have $cpl(x, c) = cpl(y, hc(c, \tilde{k}))$ with hc as defined in Equation 8.3. We thus can define the diversity measure in terms of the common prefix length in the same manner as the distance. More precisely, for $* \in \{TD, CPL\}$, the diversity $\delta_{RP-*}(y, \tilde{k}, c)$ of a coordinate c and a return address $(y, \tilde{k})$ is

$$\delta_{RP-*}(y, \tilde{k}, c) = \delta_*(y_i, hc(c, \tilde{k})). \tag{8.4}$$

In practice, u can increase the efficiency of the computation by only determining $hc(c, \tilde{k})$ up to the first element in which it disagrees with y.

We now define two possible realizations of the routing algorithm $\mathbf{R}_{node}$, namely $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$. Given the RP return address $(y, \tilde{k})$ of the destination e, $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ forward the message to the neighbor v with the lowest diversity measure $\delta_{RP-TD}(y, \tilde{k}, id(v))$ and $\delta_{RP-CPL}(y, \tilde{k}, id(v))$, respectively.

Proving Route Preservation: We now prove formally that the above return addresses preserve routes. For this purpose, we first define the notion of preserving properties of coordinates.

Definition 8.1. *Let $Q_u : \mathcal{P}(\mathbf{X}) \times \mathbf{X} \to \mathcal{P}(\mathbf{X})$ be a local function of node u in a graph $G = (V, E)$. Given a set $C \subset \mathbf{X}_V = \{v \in V : id(v)\}$ of node coordinates and a target coordinate $x \in \mathbf{X}$, Q returns a subset $C' \subset C$. A return address $(y, \tilde{k})$ for a coordinate x is said to* preserve *Q if there exists a function $Q' : \mathcal{P}(\mathbf{X}) \times \mathbf{Y} \times \tilde{\mathbb{K}} \to \mathcal{P}(\mathbf{X})$ such that for all $C \subset \mathbf{X}$*

$$Q'(C, y, \tilde{k}) = Q(C, x).$$

The notion of *route preserving (RP)* return addresses now follows if we choose the function Q to return the neighbors with the closest coordinates to a destination.

Definition 8.2. *Let*

$$\begin{aligned} &ra\colon \mathcal{P}(\mathbf{X}) \times \mathbf{X} \to \mathcal{P}(\mathbf{X}), \\ &ra(C, x) = argmin_{c \in C}\{\delta(c, x)\} \end{aligned} \tag{8.5}$$

determine the closest coordinates in a set C to a coordinate x. A return address $(y, \tilde{k})$ is called route preserving (RP) (with regard to δ) if it preserves ra.

[1] We exclude this step in Algorithm 8.3 for increased readability

[2] Note that a diversity measure is not a distance because it i) is defined for two potentially distinct sets $\mathbf{X}$ and $\mathbf{Y}$, and ii) is not symmetric.

Based Definition 8.2, we now show that Algorithm 8.3 generates RP return addresses.

Theorem 8.3. *Algorithm 8.3 generates RP return addresses with regard to the distances δ_{TD} and δ_{CPL}.*

Proof. In order to show that $(y, \tilde{k})$ preserves routes, we derive the relation between the diversity measures δ_{RP-TD} and δ_{RP-CPL}, defined in Equation 8.4, and the corresponding distances δ_{TD} and δ_{CPL}, defined in Equation 8.1 and Equation 8.2, respectively.

Let $cord(y, \tilde{k})$ denote the padded coordinate used to generate y, and let x be the coordinate without padding. In the following, we relate the distance of x and a coordinate c to the diversity measure of $(y, \tilde{k})$ and c. We have $cpl(cord(y, \tilde{k}), c) = cpl(x, c) \leq |x|$ for any node coordinate c, i.e., the common prefix length of the padded coordinate and c is at most equal to the length of the original coordinate x. The inequality $cpl(x, c) \leq |x|$ holds because a node with coordinate c with $cpl(cord(y, \tilde{k}), c) > |x|$ cannot exist in a valid embedding. More precisely, our embeddings algorithm ensures that coordinates are unique and a node v ensures that the first element of the padding does not correspond to the $|id(v)| + 1$-th element of a descendant's coordinate. Thus, we can indeed limit our evaluation to coordinates c with $cpl(cord(y, \tilde{k}), c) \leq |x|$.

We start by considering the tree distance δ_{TD}. By Equation 8.4, we have

$$\begin{aligned}\delta_{RP-TD}(y, \tilde{k}, c) &= L + |c| - 2cpl(cord(y, \tilde{k}), c) \\ &= |x| + |c| - 2cpl(x, c) + (L - |x|) \\ &= \delta_{TD}(x, c) + (L - |x|).\end{aligned}$$

Hence, diversity measure and distance only differ by a constant independent of c. Thus, any forwarding node can determine the closest coordinates to the destination in its neighborhood. Hence, Algorithm 8.3 generates RP return addresses with regard to δ_{TD}.

When utilizing the distance δ_{CPL}, we consider two coordinates c_1 and c_2. We show that i) $\delta_{CPL}(x, c_1) = \delta_{CPL}(x, c_2)$ iff $\delta_{RP-CPL}(y, \tilde{k}, c_1) = \delta_{RP-CPL}(y, \tilde{k}, c_2)$ and ii) $\delta_{CPL}(x, c_1) < \delta_{CPL}(x, c_2)$ iff $\delta_{RP-CPL}(y, \tilde{k}, c_1) < \delta_{RP-CPL}(y, \tilde{k}, c_2)$. In other words, the return address $(y, \tilde{k})$ is RP since the comparison of two coordinates with regard to their diversity to y yields the same order as the comparison of the two coordinates with regard to their distance to x.

For case i), note that by Equation 8.2 $\delta_{CPL}(x, c_1) = \delta_{CPL}(x, c_2)$ implies that $cpl(x, c_1) = cpl(x, c_2)$ and $|c_1| = |c_2|$. Because $cpl(cord(y, \tilde{k}), c_i) = cpl(x, c_i)$, we indeed have $\delta_{RP-CPL}(y, \tilde{k}, c_1) = \delta_{RP-CPL}(y, \tilde{k}, c_2)$. The converse holds analogously by Equation 8.4.

If ii) $\delta_{CPL}(x, c_1) < \delta_{CPL}(x, c_2)$, then Equation 8.2 implies that either a) $cpl(x, c_1) > cpl(x, c_2)$ or b) $cpl(x, c_1) = cpl(x, c_2)$ and $|c_1| < |c_2|$. In the first case, the claim follows as $cpl(cord(y, \tilde{k}), c_i) = cpl(x, c_i)$ and δ_{CPL} and δ_{RP-CPL} both prefer coordinates with a longer common prefix length. For the second case, the claim follows from $cpl(x, c_1) = cpl(x, c_2)$ and $cpl(cord(y, \tilde{k}), c_i) = cpl(x, c_i)$ as

$$\begin{aligned}&\delta_{CPL}(x, c_1) < \delta_{CPL}(x, c_2) \\ \iff& L - cpl(x, c_1) - \frac{1}{|x| + |c_1| + 1} < L - cpl(x, c_2) - \frac{1}{|x| + |c_2| + 1} \\ \iff& -\frac{1}{|x| + |c_1| + 1} < -\frac{1}{|x| + |c_2| + 1} \\ \iff& |x| + |c_1| + 1 < |x| + |c_2| + 1 \\ \iff& L + |c_1| + 1 < L + |c_2| + 1 \\ \iff& -\frac{1}{L + |c_1| + 1} < -\frac{1}{L + |c_2| + 1} \\ \iff& \delta_{CPL}(cord(y, \tilde{k}), c_1) < \delta_{CPL}(cord(y, \tilde{k}), c_2) \\ \iff& \delta_{RP-CPL}(y, \tilde{k}, c_1) < \delta_{RP-CPL}(y, \tilde{k}, c_2)\end{aligned}$$

Hence, Algorithm 8.3 generates RP return addresses with regard to δ_{CPL} as well. □

Up to now, we have only considered route preserving return addresses generated by padding coordinates and applying a hash cascade. Optionally, an additionally layer of symmetric encryption can be added, preventing a node v from deriving the actual length of the common prefix. Rather, v can only determine if a neighbor is closer to the destination than v itself. However, the additional layer reduces the efficiency as nodes select one closer neighbor at random rather than the closest neighbor and it remains unclear to what extent such a layer indeed improves the anonymity. For this reason, the advantage of the additional layer is limited, so that we focus on RP return addresses here and defer the further obfuscation of coordinates to Appendix E.

We prove that Algorithm 8.3 indeed enables receiver anonymity in Section 8.5.

8.2.4 Content Storage

Tree embeddings do not allow balanced content addressing without either weakening the resilience or requiring prohibitive stabilization complexity, as shown in Chapter 7. In contrast, virtual overlays provide balanced content addressing by design but make use of tunnels that cannot be stabilized efficiently, as shown in Chapter 6. Here, we suggest replacing the tunnels through routing in greedy embeddings. Note that we only sketch the solution for content storage and retrieval because our focus lies on improving the quality of the greedy embeddings for messaging between nodes. In the following, we first present the idea of our design and then a realization based upon a recursive Kademlia.

General Design: Nodes establish a DHT by maintaining a routing table of (virtual) overlay connections. The routing table contains entries consisting of a DHT coordinate and return addresses of the corresponding node. Nodes communicate with their virtual neighbors by sending requests in any of the γ embedding.

New routing table entries are added by routing for a suitable virtual overlay key, as done in [110] for the tunnel discovery. However, after the routing terminates, the discovered nodes send back their return addresses rather than using the routing path as a new tunnel. In this manner, the length of routes between virtual overlay neighbors only depends on the embeddings and does not increase over time under the assumption that the tree depth remains mostly unaffected by temporal changes. The exact nature of the neighbor discovery, the routing algorithm $\mathbf{R}_{content}$, and the stabilization of the virtual overlay depend on the specifications of the DHT.

Kademlia: In our evaluation, we utilize a highly resilient recursive Kademlia [78]. In Kademlia, a node selects a *Kademlia Identifier* $ID(v)$ uniformly at random in the form of a 160-bit number. The distance between identifiers is equal to their XOR. Nodes maintain many redundant (virtual) overlay connections to increase the resilience. More precisely, each node v keeps a *routing table* of k-buckets. The j-th bucket contains up to k addresses of nodes u such that the common prefix length of $ID(v)$ and $ID(u)$ is j. Maintaining more than 1 neighbor per common prefix length increases the robustness to failures and possibly even to attacks due to the existence of alternative connections.

Based on such routing tables, efficient and robust content discovery is possible. Files are indexed by keys corresponding to the hash of their content, i.e., the algorithm $\mathbf{AdGen}_{content}$ for the generation of file addresses is a hash function. A node u requesting a file with key f looks up the closest nodes $v_1, \ldots, v_\alpha$ to f in its routing table in terms of virtual overlay coordinates. Then, u routes for each v_i in τ trees. Upon receiving the request, v_i returns f via the same route if in possession of f. If v_i has already received the request via a parallel query, v_i sends a backtrack message such that u can contact a different node. Otherwise, v_i forwards the message to the virtual overlay neighbor closest to f, again using tree routing, and sends an acknowledgment message to u. If a node does not receive an acknowledgment from its overlay neighbor in time, it selects an alternative node from its routing table if virtual neighbors closer to f than u exist. In this manner, the routing algorithm can deal with failures and dropped messages.

Stabilization is realized in the same reactive manner as in the original Kademlia. Whenever a node u successfully sends a message to an overlay neighbor v, v returns an acknowledgment containing updated return addresses if any coordinates have changed. In addition, if forwarding the request, it returns the routing table entry of the selected next hop. u can add the entry to its table if there is an empty slot in the respective bucket. If any overlay neighbor in the routing table does not acknowledge a request within a certain time, u removes the neighbor from the routing table. Depending on the implementation, u initializes a new neighbor discovery request. In this manner, nodes integrate stabilization in the routing process.

We have now presented the essential components of our design. In the following, we evaluate our design with regard to our requirements.

8.3 Efficiency and Scalability

In this section, we analyze the efficiency of our scheme with regard to routing complexity, stabilization complexity, and their evolution over time, following the methodology discussed in Section 5.2. We start with a theoretical analysis in order to prove that our algorithms achieve the desired asymptotic bounds. Afterwards, we present the results of our simulation-based evaluation.

8.3.1 Theoretical Analysis

In the first part of this section, we obtain upper bounds on the expected routing length, as defined in Equation 5.6, of the routing algorithms $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$. The desired upper bound on the routing complexity, as defined in Equation 5.5, follows by multiplying this bound for routing in one tree with τ, the number of trees used for parallel routing. Afterwards, we consider the stabilization complexity $CS^{\mathbf{S}}$ of the stabilization algorithm $\mathbf{S}$ consisting of i) the local reconstruction of the trees according to Algorithm 8.1 and ii) the assignment of new coordinates for the nodes affected by a change topology using the modified PIE embedding. Last, we determine the long-term behavior of routing and stabilization complexity following the model presented in Section 5.2.3.

Routing: We consider both messaging between nodes as well as content discovery in the DHT.

Theorem 8.4. *Let id be a modified PIE embedding on a spanning tree of G generated by Algorithm 8.1 with parameters γ and q. Furthermore, assume that the diameter of G is $diam(G) = \mathcal{O}(\log n)$. The expected routing length of Algorithm 8.2 is at most*

$$\mathbb{E}(R^{TD}) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right) \tag{8.6}$$

for the routing algorithm $\mathbf{R}^{TD}$, and

$$\mathbb{E}(R^{CPL}) = \mathcal{O}\left(\left(\frac{\gamma}{q}\right)^2 \log n\right) \tag{8.7}$$

for $\mathbf{R}^{CPL}$.

For the proof, we first show Lemma 8.5, which bounds the expected level of a node in trees constructed by Algorithm 8.1. More precisely, we prove that the expected level of a node in any tree constructed by Algorithm 8.1 is increased by at most a constant factor in comparison to a breath-first-search.

Lemma 8.5. *Let T be any of the γ trees constructed by Algorithm 8.1 and r the root of T. Furthermore, denote by $sp_r(v)$ the length of the shortest path from v to r, and let $L_T(v)$ be the level of v in T. Then the expected value of $L_T(v)$ is bound by*

$$\mathbb{E}(L_T(v)) \leq sp_r(v) \cdot \left(1 + \frac{\gamma}{q}\right). \tag{8.8}$$

Proof. Note that the number of rounds until a node accepts an invitation is an upper bound on its depth. We thus first give an upper bound on the expected number of rounds until a node v accepts an invitation for T after receiving the first invitation. Afterwards, we show Equation 8.8 by induction.

In the first step, we denote the number of rounds until acceptance by Y. In order to derive an upper bound on $\mathbb{E}(Y)$, we assume that v does not receive any invitation that it can immediately accept, i.e., an invitation from neighbors u with minimal parent count $pc(u)$. Thus, v accepts one invitation with probability q in each round. In the worst case, the γ-th accepted invitation is for tree T. The number of rounds thus corresponds to the sum of γ identically distributed geometrically distributed random variables $X_1, \ldots, X_\gamma$. Here, X_i is the number of trials until the first success of a sequence of Bernoulli experiments with success probability q, i.e., the number of rounds until an invitation is accepted. The random variable $X = X_1 + \ldots + X_\gamma$ describes the number of trials until the γ-th success and presents an upper bound on the expected number of rounds until acceptance of an invitation for tree T. We hence derive an upper bound on $\mathbb{E}(Y)$ by

$$\mathbb{E}(Y) \leq E(X) = \sum_{i=1}^{\gamma} \mathbb{E}(X_i) = \gamma\mathbb{E}(X_1) = \frac{\gamma}{q}. \tag{8.9}$$

In the second step, we apply induction on $l = sp_r(v)$. For $l = 1$, the node v receives an invitation from r at round 1 of the protocol because v is a neighbor of the root node. In expectation, v joins T at round at most $1 + \mathbb{E}(Y) \leq 1 + \frac{\gamma}{q}$, which shows the claim for $l = 1$. Now, we assume Equation 8.8 holds for $l - 1$ and show that then it also holds for l. The number of rounds Z until the node v with $sp_r(v) = l$ accepts an invitation in tree T is the sum of Z_1, the number of rounds until the first invitation is received, and Z_2 the number of rounds v accepts after receiving the first invitation. v is the neighbor of a node w with

$sp_r(w) = l - 1$ and receives an invitation from w one round after w joined T. So, Z_1 is bound by our induction hypothesis, and Z_2 is equal to Y and hence bound by Equation 8.9. As a result,

$$\mathbb{E}(Z) = \mathbb{E}(Z_1) + 1 + \mathbb{E}(Z_2) \leq (l-1) \cdot \left(\frac{\gamma}{q} + 1\right) + 1 + \frac{\gamma}{q} + 1 = l \cdot \left(\frac{\gamma}{q} + 1\right),$$

and hence indeed Equation 8.8 holds. □

Based on Lemma 8.5, we now prove Theorem 8.4. The idea of the proof is to bound the routing length by a multiple of the expected level of a node.

Proof. We consider the diversity measure δ_{RP-TD} first and then δ_{RP-CPL}.

For δ_{RP-TD}, the claim follows directly from Lemma 8.5 and Theorem 4.3 in [80]. More precisely, the expected level of a node is at most $\mathcal{O}\left(\frac{\gamma}{q} \log n\right)$ assuming a diameter and hence maximal distance to the root of $\mathcal{O}(\log n)$. Recall that the distance $\delta_{TD}(id(s), id(e))$ of two nodes s and e corresponds to the length of the shortest path between them in the tree and is an upper bound on the routing. Now, by Equation 8.1, the sum of the length of the two coordinates is an upper bound on $\delta_{TD}(id(s), id(e))$. As the length of a coordinate is equal to the level of the corresponding node in the tree, we indeed obtain

$$\mathbb{E}(R_{s,e}^{TD}) \leq \mathbb{E}(\delta_{TD}(id(s), id(e))) \leq \mathbb{E}(L_T(s)) + \mathbb{E}(L_T(e)) = \mathcal{O}\left(\frac{\gamma}{q} \log n\right). \tag{8.10}$$

The last step follows from Lemma 8.5. Equation 8.6 follows because Equation 8.10 holds for all source-destination pairs (s, e).

In contrast, the proof for the distance δ_{CPL} cannot build on previous results. Note that the change of the distance function does not affect the existence of a path with expected length at most $\mathbb{E}(L_T(s)) + \mathbb{E}(L_T(e))$ between source s and destination e in the tree. However, the routing might divert from that path when discovering a node with a longer common prefix length but at a higher depth. For this reason, the sum of the expected levels is not an upper bound on the routing length. Rather, whenever a node with a longer common prefix length is contacted, the upper bound of the remaining number of hops is reset to the expected level of that node in addition to the level of e. In the following, we show that such a reset increases the distance in the tree by less than $\frac{\gamma}{q}$ on average. The claim then follows because the number of resets is bound by the expected level of the destination. Equation 8.7 follows by multiplication of the increased distance per reset and the number of resets.

Formalizing the above, let X_i give the tree distance between the i-th contacted node v_i and the target e. As $(X_i)_{i \in \mathbb{N}_0}$ is not a monotonously decreasing process, we cannot use the methodology for such processes. Rather, we need to bound the number of times Z_1 that X_i increases and the expected amount of increase Z_2. Thus, the routing length $R_{s,e}^{CPL}$ from a source node s to e is bound by

$$\mathbb{E}(R_{s,e}^{CPL}) \leq \mathbb{E}(L_T(s)) + \mathbb{E}(L_T(e)) + \mathbb{E}(Z_1)\mathbb{E}(Z_2). \tag{8.11}$$

The number of times Z_1 the common prefix length can increase is bound by the length of the target's coordinate and hence its level in T. So by Lemma 8.5,

$$\mathbb{E}(Z_1) \leq \mathbb{E}(L_T(e)). \tag{8.12}$$

The tree distance X_i is potentially increased whenever a node with a longer common prefix length is contacted. Yet, an upper bound on the expected increase is given by the difference in the levels $L_T(v_i)$ and $L_T(v_{i+1})$ minus 1 due to the increased common prefix length. Note that v_i and v_{i+1} are neighbors and hence the length of their shortest path to the root differs by at most 1. Lemma 8.5 thus provides the desired bound on $\mathbb{E}(Z_2)$

$$\mathbb{E}(Z_2) \leq \mathbb{E}\left(L_T(v_i) - L_T(v_{i+1})\right) - 1 = \frac{\gamma}{q}. \tag{8.13}$$

The desired bound can now be derived from Lemma 8.5, Equations 8.11, 8.12, and 8.13 under the assumption that the diameter of the graph and hence all shortest paths to the root scale logarithmically, i.e.,

$$\mathbb{E}(R_{s,e}^{CPL}) \leq \mathbb{E}(L_T(s)) + \mathbb{E}(L_T(e)) + \mathbb{E}(L_T(e))\frac{\gamma}{q} = \mathcal{O}\left(\left(\frac{\gamma}{q}\right)^2 \log n\right). \tag{8.14}$$

As for the first part, Equation 8.7 follows because Equation 8.14 holds for all pairs (s, e). □

The bounds for a virtual overlay lookup based on routing algorithm $\mathbf{R}_{content}$ follow directly from the fact that a DHT lookup requires $\mathcal{O}(\log n)$ overlay hops with each hop corresponding to one route in the network embedding.

Corollary 8.6. *If the DHT used for the virtual overlay offers logarithmic routing, the communication complexity of routing algorithm* $\mathbf{R}_{content}$ *is*

$$\mathbb{E}(DHT^{TD}) = \mathcal{O}\left(\frac{\gamma}{q}\log^2 n\right)$$

for the diversity measure δ_{RP-TD} *and*

$$\mathbb{E}(DHT^{CPL}) = \mathcal{O}\left(\left(\frac{\gamma}{q}\right)^2 \log^2 n\right)$$

for diversity measure δ_{RP-CPL}.

Stabilization: The stabilization complexity is required to stay polylog in the network size to allow for scalable communication and content addressing. In the following, we hence give bounds for the stabilization of the network embeddings, the complexity for the virtual overlay follow by considering the stabilization of the DHT as suggested for general overlay networks and multiplying with the length of the routes between overlay neighbors.

Theorem 8.7. *We assume the social graph* G *to be of a logarithmic diameter and a constant average degree. Furthermore, we assume the use of a the root election protocol with complexity* $\mathcal{O}(n \log n)$. *Then the stabilization complexity* $CS^{\mathbf{S}}$ *of the spanning trees constructed by Algorithm 8.1 with parameters* γ *and* q *is*

$$\mathbb{E}(CS^{\mathbf{S}}) = \mathcal{O}\left(\gamma\frac{\gamma}{q}\log n\right). \tag{8.15}$$

Proof. We first consider the complexity for one tree. The general result then follows by multiplying with the number of trees γ. When a node joins an overlay with a constant average degree, the expected communication complexity of receiving and replying to all invitations is constant. For a node departure, we consider non-root nodes and root nodes separately. If any node but the root departs, the expected stabilization complexity corresponds to the number of nodes that have to rejoin T. This number of nodes is equal to the number of descendants in a tree. Hence, the expected complexity of a departure corresponds to the expected number of descendants. Consider that a node on level l is a descendant of l nodes, so that the expected number of descendants D is given by

$$\mathbb{E}(D) = \frac{1}{|V|}\sum_{v\in V}\mathbb{E}(L_T(v)) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right).$$

If the root node leaves, the spanning tree and the embedding have to be re-established at a complexity of $\mathcal{O}(n \log n)$. As the probability for the root to depart is $1/n$, we indeed have

$$\mathbb{E}(CS^{\mathbf{S}}) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right) + \mathcal{O}\left(\frac{1}{n}n\log n\right) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right).$$

□

Combined Dynamic Analysis: Last, we show that the routing and stabilization complexity remain stable over an extended period of time, based on the model of subsequent joins and departures in Section 5.2.3. The result follows as the stabilization executes the tree construction for the affected nodes, simulating the rounds in order to maintain the same guarantees on the depth.

Theorem 8.8. *Let* $\left(R_t^{\mathbf{R}^{TD}}\right)_{t\in\mathbb{N}_0}$ *and* $\left(R_t^{\mathbf{R}^{CPL}}\right)_{t\in\mathbb{N}_0}$ *denote the routing length over time with respect to the diversity metrics* δ_{RP-TD} *and* δ_{RP-CPL}. *The expected routing length is*

$$\mathbb{E}\left(R_t^{\mathbf{R}^{TD}}\right) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right),$$

and

$$\mathbb{E}\left(R_t^{\mathbf{R}^{CPL}}\right) = \mathcal{O}\left(\left(\frac{\gamma}{q}\right)^2 \log n\right),$$

respectively. Furthermore, let $\left(CS_t^{\mathbf{S}}\right)_{t\in\mathbb{N}_0}$ *be the complexity of the tree stabilization. Under the assumption of a root election protocol with complexity* $\mathcal{O}(n \log n)$*, the expected stabilization complexity is*

$$\mathbb{E}\left(CS_t^{\mathbf{S}}\right) = \mathcal{O}\left(\gamma \frac{\gamma}{q} \log n\right).$$

Proof. Note that both the routing length and the stabilization complexity depend on the average depth of a node in the trees. We show in Lemma 8.5 that Algorithm 8.1 constructs trees of an expected mean depth of at most $\left(1 + \frac{\gamma}{q}\right) \log n$. As Algorithm 8.1 does not necessarily assume a parallel construction of trees, the bound also holds if only the affected trees are rebuilt. Hence, the results from Theorem 8.4 and Theorem 8.7 remain valid for the dynamic system. □

We have shown that the communication complexity is polylog as required. Furthermore, we achieve balanced content addressing as by the respective results for DHTs [97]. Thus, we achieve all our requirements from Section 2.6 with regard to scalability.

8.3.2 Simulations

In this section, we validate the above bounds and relate them to the concrete communication overhead for selected scenarios. We start by detailing our simulation model and set-up, followed by our expectations, the results and their interpretation.

Model and Evaluation Metrics: We integrated the embedding and routing algorithms into the simulation model described in Chapter 5. In order to quantify our improvements, we compared our results to those for Freenet, a virtual overlay *VO*, and the original PIE embedding. We described Freenet, MCON, and PIE in Sections 3.2, 3.3, and 7.4, respectively, and implemented them accordingly.

As the success ratio of Freenet was extremely low when using a *htl* counter, our results are based upon a variant of Freenet without such a counter, which guarantees the eventual node discovery.

The virtual overlay *VO* combines the advantages of X-Vine and MCON by using shortest paths as tunnels in a Kademlia overlay like MCON but integrating backtracking in the presence of local optima and shortcuts from one tunnel to another like X-Vine.

For the evaluation, we sampled the routing length and the stabilization complexity as the average of multiple runs with m routes or topology changes each, following the procedure described in Section 5.2.4. As new nodes can be integrated into the trees at a constant overhead, we only considered the overhead of node departures, namely the number of descendants of the departing node in all trees combined. Note that the actual number of messages required for stabilization is slightly higher because the departing nodes needs to inform its remaining neighbors. Furthermore, the descendants need to exchange multiple messages for re-joining the trees. However, the actual number of messages depends on the details of the implementation. In order to provide an abstract bound, the number of descendants presented a natural implementation-independent metric, as Section 8.3.1 shows that it is the dominating factor with regard to the stabilization complexity.

Note that we only make use of the snapshot-based evaluation, because routing and stabilization complexity remain largely unaffected by dynamics, as can be seen from Theorem 8.8.

Set-up: We restrict our presentation to the Facebook graph *FB*. Section 4.2 provides an overview of *FB*'s important topological features. Appendix F presents additional results, in particular comparing different social network topologies.

The spanning tree construction in Algorithm 8.1 is parametrized by the number of trees $\gamma \in \{1, 2, 3, 5, 7, 10, 12, 15\}$, the acceptance probability $q = 0.5$, and the selection criterion W chosen to be either random selection (denoted *DIV-RAND*) or preferred selection of nodes at a low depth (denoted *DIV-DEP*). In addition, we consider a breadth first search for spanning tree construction (denoted *BFS*). Moreover, we consider the impact of the two distances δ_{TD} (denoted *TD*) and δ_{CPL} (denoted *CPL*). The length of the return addresses was set to $L = 128$ and the number of bits per element was $b = 128$, all $\tau = \gamma$ embeddings were considered for routing.

For the virtual overlay used for content addressing, we chose a highly resilient recursive Kademlia [78] with bucket size $k = 8$ and $\alpha \in \{1, 3\}$ parallel look-ups. Because routing table entries are not uniquely determined by Kademlia identifiers, the entries were chosen randomly from all suitable candidates.

We parametrized the related approaches as follows. For simulating Freenet, we executed the embedding for 6,000 iterations as suggested in [142] and then routed using a distance-directed depth-first search based only on the information about direct neighbors. The routing and stabilization complexity of the original PIE embedding is equal to the respective quantities of our algorithm for $\gamma = 1$, the distance function δ_{TD} and routing without the use of backtracking. In order to better understand the results of the comparison, we simulate the virtual overlay VO using the same Kademlia overlay as for our own approach but replacing the tree routing by tunnels corresponding to the shortest paths between overlay neighbors. So, we parametrized the related approaches by either using the proposed standard parameters or selecting parameters that are suitable for comparison because they corresponds to the same degree of redundancy as the parametrization of our own approach.

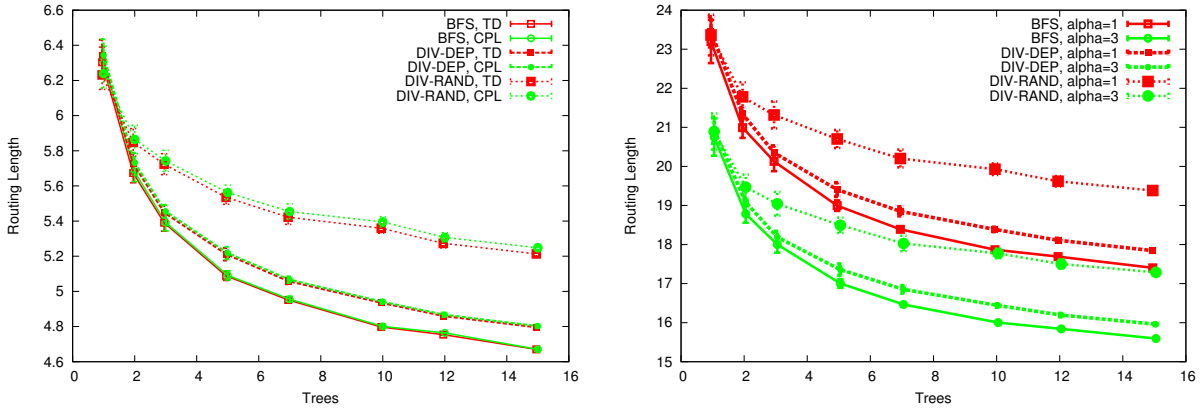
All results were averaged over 20 runs. They are displayed with 95% confidence intervals. Each run consisted of 100,000 routing attempts for a randomly selected source-destination pair.

Expectations: We expect that the routing length decreases with the number of embeddings, because the number of available routes and thus the probability to discover the shortest route in one embedding increases. In general, the routing length is directly related to the tree depth and should thus be lower for *BFS* and *DIV-DEP*.

Similarly, we expect a higher stabilization overhead for trees of a higher depth as the expected number of descendants per node increases. Thus, the number of nodes that need to select a new parent should be higher for *DIV-RAND* than for *DIV-DEP* and *BFS*.

In comparison to the existing approaches, our approach should enable shorter routes between pairs of nodes than both Freenent and VO. As shown above, we achieve a routing complexity of $\mathcal{O}(\log n)$ whereas the related work achieves at best routes of polylog length. However, our routes for content discovery should be slightly longer than in VO. VO utilizes the same DHT routing but uses the shortest paths rather than the longer tree routes.

Results: In the following, we first give some results with regard to the depth of the constructed trees, as the tree depth is essential for both the routing and the stabilization complexity. Afterwards, we present our results for the communication overhead, subsequently considering the routing algorithms $\mathbf{R}_{node}$, $\mathbf{R}_{content}$, and the stabilization algorithm $\mathbf{S}$. Last, we present our comparison to the state-of-the-art approaches.



(a) $\mathbf{R}_{node}$ (b) $\mathbf{R}_{content}$

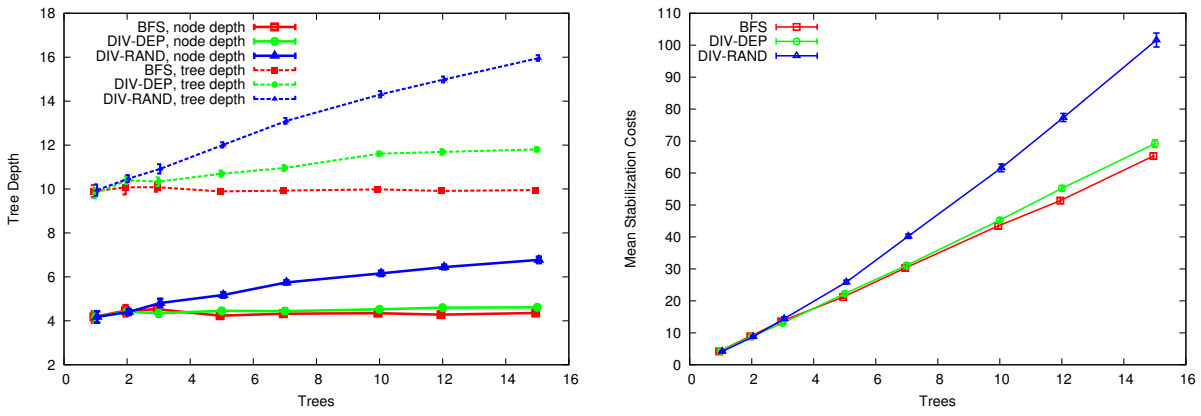
Figure 8.4: Impact of number of embeddings γ, tree construction, and distance function on routing length for a) tree routing and b) Kademlia lookup with degree of parallelism α; related approaches result in routing lengths of 14 (virtual overlay VO) and close to 10,000 (Freenet)

We start by considering the depth of the trees, as Section 8.3.1 indicates that the average level/depth of nodes in the tree is of critical importance for both the routing length and the stabilization complexity. Figure 8.5a displays both the depth of a node, averaged over all nodes and γ trees, as well as the tree depth, i.e., the maximal depth of a node in a tree averaged over γ spanning trees. *BFS* constructed each tree independently regardless of the maximal number of trees, so that the average depth remained constant in γ (apart from insignificant deviations due to the probabilistic nature of the root election) with a depth of slightly above 4 for nodes and roughly 10 for trees. In contrast, *DIV-DEP*, which aims to select diverse parents but prefers parents close to the root, exhibited a slow increase in the depth. When the number of trees and hence the probability to select an alternative parent at a higher depth increased,

the depth of nodes increased from 4.17 ($\gamma = 1$) to 4.61 ($\gamma = 15$) while the tree depth increased from 9.8 to 11.8. For *DIV-RAND*, the increase was pronounced, seemingly close to linear in γ, with average depths of 6.77 (nodes) and 15.96 (trees) for $\gamma = 15$. We now show how differences with regard to the depths of the tree directly translate to differences in routing length and stabilization overhead.

The impact of the three parameters, number of trees, tree construction, and distance, on the routing length of $\mathbf{R}_{node}$ confirms our expectations. First, the results indicate that the tree construction, in particular the number of trees, is the dominating factor for the routing length. So, the routing length decreased considerably if multiple embeddings were used because the shortest route in any of the trees was considered. Second, preferring parents closer to the root, i.e., using *BFS* or *DIV-DEP*, produced shorter routes in the tree and hence reduced the routing length. Third, in comparison to the tree construction, the choice of a distance function had less impact. For *BFS* or *DIV-DEP*, the advantage of *TD* over *CPL* was barely noticeable, whereas the difference for *DIV-RAND* was still small but noticeable. In order to understand this difference, note that *CPL* is expected to lead to longer routes. The reason for the longer routes lies in forwarding the request to neighbors at a higher depth, which might have a long common prefix but are nevertheless at a higher distance from the destination due to their depth. For *BFS* or *DIV-DEP*, the difference of the depth of neighbors was generally small because neighbors at a lower depth were preferably selected as parents. In contrast, *DIV-DEP* allows for larger differences in the depth of neighbors. Hence, there is a higher probability to increase the tree distance by selecting a neighbor with a longer common prefix length but at a high depth. All in all, the routing length varied between 4.67 (*BFS*, $\gamma = 15$, *TD*) and 6.24 (*DIV-RAND*, $\gamma = 1$, *CPL*) hops, as displayed in Figure 8.4a. In summary, the use of multiple embeddings indeed reduced the routing length considerably.

The performance of the DHT lookup $\mathbf{R}_{content}$ in the virtual overlay directly related to the previous results (cmp. Fig. 8.4b for the distance under *TD*). The overhead for the discovery of a randomly chosen Kademlia ID, stored at the node with the closest ID in the overlay, varied between 15.56 and 24.25 hops in the F2F overlay, at around 4 hops in the virtual overlay.



(a) Tree Depth (b) Stabilization

Figure 8.5: a) Depth of Spanning Trees: average depth (level) of a node/tree for different tree construction algorithms, directly corresponds to routing length and stabilization overhead; b) Stabilization overhead varying number of trees and tree construction, in contrast the virtual overlay *VO* needs at least 10,000 messages for stabilization

By Theorem 8.7, the stabilization complexity was expected to increase at most quadratic with the number of trees. Indeed, Figure 8.5b supports this fact for *DIV-RAND*. The increase for *BFS* and *DIV-DEP* was even only linear and slightly super-linear, respectively. Note that the quadratic increase is due to the raising average depth of additional trees generated by Algorithm 8.1. With the goal of achieving diverse spanning trees, nodes select parents at a higher depth. However, the average number of descendants increases with the depth, because a node at depth l is a descendant of l nodes. Due to the stabilization complexity corresponding to the number of the departing node's descendants, the stabilization overhead was higher for *DIV-RAND* and *DIV-DEP* than for *BFS*. More precisely, *BFS* constructs all γ trees independently, so that the average depth of each tree is independent of the number of trees. The stabilization complexity per tree thus remains constant. *DIV-DEP*, aiming to balance diversity and short routes, causes stabilization overhead between the two former approaches, but performed closer to *BFS* (this similarity also held for the routing length). More concretely, the average stabilization overhead for a departing node was slightly below 4.5 for a single tree. For $\gamma = 15$, it increased to 65 (*BFS*), 69 (*DIV-DEP*), and more than 101 (*DIV-RAND*). In contrast to a complete re-computation of

the embedding requiring at least $n = 63392$ messages, the stabilization overhead is negligible.

For the related approaches, we found a routing length of 9403.1 for Freenet, 16.11 for VO with $\alpha = 1$, and 14.07 for VO with $\alpha = 3$. Furthermore, the shortest paths are on average of length 4.31, meaning that our routing length of 4.67 is close to optimal. So, routing between nodes in the tree required less than half the overhead of state-of-the-art approaches. Routing in the virtual overlay, requiring at best less than 16 hops in our scheme, was slightly more costly in our approach than in VO due to the inability of the tree routing to guarantee shortest paths between virtual neighbors.

A straight-forward comparison of the stabilization overhead was not possible. Since Freenet stabilizes periodically, there is no overhead directly associated with a leaving node. In case of virtual overlays, VO uses flooding for stabilization, which is clearly more costly. Other overlays such as X-Vine use less costly stabilization but stabilization and routing overhead are unstable and increase over time as shown in Chapter 6, so that it is unclear which state of the system should be considered for a comparison. In order to nevertheless give a lower bound on the stabilization overhead, we computed the number of tunnels that needed to be rebuild in VO. On average, 477.35 tunnels corresponding to the shortest paths were affected by a departing node. If a tunnel is repaired by routing in the Kademlia overlay like in X-Vine, the stabilization overhead per tunnel corresponds to routing a request and the respective reply, i.e., for tunnels corresponding to the shortest paths at least $2 \cdot 14 = 28$ messages, resulting in a lower bound on more than $10,000$ messages per node departure. As shown in Chapter 6, the above stabilization algorithm is unable to maintain short routes, such that the actual overhead of stabilization in virtual overlay is even higher than the above lower bound.

Discussion: Our simulation study validates the asymptotic bounds. Indeed, the routing length and thus the routing complexity for messaging is very low, improving on the state-of-the-art by more than a factor of 3. The stabilization complexity is similarly low if the number of trees is not too high. Even for $\gamma = 15$ trees, the number of involved nodes is generally well below 100, which still improves upon virtual overlays such as VO, the most promising state-of-the-art candidate. Only content discovery in form of a DHT lookup was slightly more costly in our approach than in VO, which we consider acceptable given the considerable advantage with regard to all other metrics.

We have considerably improved the efficiency of F2F overlays. In the following, we show that we also mitigated their vulnerability to failures and attacks.

8.4 Robustness and Censorship-Resistance

In this section, we consider the robustness and resistance to censorship of VOUTE. We follow the methodology from Section 5.4. Note that the evaluation of the censorship-resistance requires a specification of the modified stabilization algorithm $\mathbf{S}'$, which we refer to as *attack strategy* in the following. After deriving two attack strategies, we subsequently present our theoretical and simulation-based evaluation.

We express our results in terms of node coordinates and distances δ_{TD} and δ_{CPL} rather than the corresponding return addresses and diversity measures. The use of distances simplifies the notation as we do not need to apply a hash cascade for the comparison of coordinates and return addresses. As the routing paths are chosen identical for both coordinates and return addresses, the results are equally valid for return addresses.

8.4.1 Attack Strategies

In order to analyze the attack resistance based on the methodology in Section 5.4, we design realizations of the attack strategies $\mathbf{S}'$ determining the attacker behavior. We first describe our attack strategies and then comment on additional strategies and our reasons on choosing these strategies. In order to model secure and insecure root selection protocols, we consider two strategies *ATT-RAND* and *ATT-ROOT*. In the following, assume that one attacker node has established x links to honest nodes and now aims to censor communication.

For secure spanning trees, the adversary A is unable to manipulate the root election. Nevertheless, A can manipulate the subsequent embedding. The attack strategy *ATT-RAND* assigns each of its children a different random prefix rather than the correct prefix. In this manner, routing fails because nodes in the higher levels of the tree do not recognize the prefix. So, the impact of the attack is increased in comparison to a random failure.

In contrast, if the adversary A can manipulate the root election protocol, *ATT-ROOT* manipulates the root election in all spanning trees such that A becomes the root in all trees. Under the assumption

that the root observes the maximal number of requests, the attack should result in a high ratio of failed requests.

Now, we shortly comment on some further attack strategies we choose not to implement and give reasons for our decision not to do so.

First, note that in the original PIE embedding, assigning the same coordinate to two children is another attack strategy. In contrast to the above strategy, the routing can then fail even if the attacker is not involved in forwarding the actual request: node coordinates are not unique and thus the request might end up at a different node than the receiver. In the modified embedding, the child decides on the last element of the coordinate. Hence, the attacker can only assign a node w the coordinate of another node v as a prefix, so that the two nodes appear to be close but are indeed not. However, upon realizing that w does not offer a route to v, the routing algorithm backtracks, so that this attack strategy merely increases the routing complexity but does not reduce the success ratio. Thus, we do not consider it here.

Second, recall from Section 5.4 that the attacker can also generate an arbitrary number of identities whereas the above attack strategies only rely on one identity. In the following, we argue that without additional knowledge, the use of additional identities in the tree does not improve the strength of the attack.

ATT-RAND actually simulates different virtual identities by providing fake distinct prefixes to all children. Indeed, in practice, it might be wise to indeed use distinct physical nodes because it minimizes the risk of detection if two neighbors realize that they are connected to the same physical node but received different prefixes.

For *ATT-ROOT*, the attacker might have to create (virtual) identities in order to manipulate the root election. As soon as A is the root in each tree, multiple identities could be used to provide prefixes of different lengths to neighbors. However, if a neighbor u of A receives a long prefix from A, there is a high chance that u and potential descendants of u choose different parents seemingly closer to the root. Thus, in expectation a large number of nodes joins those subtrees rooted at a neighbor of A with a short prefix. As routing within such a subtree does not require forwarding a request from A, A's impact is likely to be reduced. Hence, without concrete topology knowledge, the insertion of additional virtual identities (corresponding to prefixes of different lengths) does not present an obvious advantage for A.

An alternative strategy to undermine the embedding process would be to prevent the tree construction by e.g., constantly simulating a failure of the root. However, we initially excluded such pollution attacks due to the existence of protection schemes. For instance, nodes could refrain from accepting more than a certain number of roots within a time interval in order to guarantee the convergence of the tree construction. Thus, we focus on attack strategies that maximize the attacker's damage without undermining the convergence of construction.

8.4.2 Theoretical Analysis

We present two theoretical results in this section. First, we characterize the backtracking algorithm more closely. Second, we show that the censorship-resistance is improved by using the distance δ_{CPL} rather than δ_{TD}.

Throughout this section, let $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ denote Algorithm 8.2 with distance δ_{TD} and δ_{CPL}, respectively. Furthermore, let $\mathbf{GR}^{TD}$ and $\mathbf{GR}^{CPL}$ denote the corresponding standard greedy routing algorithms, which terminate in local optima with regard to the distance to the destination's coordinate. Recall from Section 5.4 that $Succ^{\mathbf{R}}$ denotes the success ratio of a routing algorithm $\mathbf{R}$, defined as the average over the random variables $Succ^{\mathbf{R}}_{s,e}$ indicating the fact that routing from a source node s to a destination e is successful. We are considering the success ratio for one embedding. The overall success ratio is improved as it is the combined success ratio of all embeddings.

Lemma 8.9. *We have that*

$$\begin{aligned} \mathbb{E}\left(Succ^{\mathbf{R}^{TD}}\right) &\geq \mathbb{E}\left(Succ^{\mathbf{GR}^{TD}}\right) \\ \mathbb{E}\left(Succ^{\mathbf{R}^{CPL}}\right) &\geq \mathbb{E}\left(Succ^{\mathbf{GR}^{CDF}}\right) \end{aligned}. \tag{8.16}$$

Furthermore, Algorithm 8.2 is successful if and only if there exists a greedy path of responsive nodes according to its distance metric δ.

Proof. Equation 8.16 follows because Algorithm 8.2 is identical to the standard greedy algorithm until the latter terminates. Then, Algorithm 8.2 continues to search for an alternative, possible increasing the success ratio.

For the second part, recall that a greedy path is a path $p = (v_0, \ldots, v_l)$ such that the distance to the destination v_l decreases in each step, i.e., $\delta(id(e), id(v_i)) < \delta(id(e), id(v_{i-1}))$ for all $i = 1, ..., l$ and a distance δ. Assume Algorithm 8.2 does not discover a route from the source $v_0 = s$ and $v_l = e$ despite the existence of a greedy path $p = (v_0, v_1, \ldots, v_{l-1}, v_l)$ of responsive nodes. Let V_R be the set of nodes that forwarded the request according to Algorithm 8.2, and let $j = \max\{i : v_i \in V_R\}$. Then the neighbor of v_{j+1} did not receive the request despite being closer to e than v_j. Though v_j might have a neighbor w closer to e than v_{j+1}, the request is backtracked to v_j if forwarding to w does not result in a route to the destination. Routing only terminates if either a route is found or v_j has forwarded the request to all closer neighbors, including v_{j+1}. Thus, Algorithm 8.2 cannot fail if a greedy path exists. In contrast, if there are no any greedy paths from s to e, any path $p = (v_0, v_1, \ldots, v_{l-1}, v_l)$ with $v_0 = s$ and $v_l = e$ contains a pair (v_{i-1}, v_i) with $\delta(id(e), id(v_i)) \geq \delta(id(e), id(v_{i-1}))$. Thus, Algorithm 8.2 does not forward the request to v_i and hence does not discover a path from s to e. It follows that indeed Algorithm 8.2 is successful if and only if a greedy path of responsive nodes exists. □

Now, we apply Lemma 8.9 to show that the use of δ_{CPL} generally enhances the censorship-resistance.

Theorem 8.10. *Let A be an attacker applying either* ATT-RAND *or* ATT-ROOT. *Then for all distinct nodes $s, e \in V$*

$$Succ_{s,e}^{\mathbf{R}^{CPL}} = 0 \implies Succ_{s,e}^{\mathbf{R}^{TD}} = 0, \tag{8.17}$$

i.e., if $\mathbf{R}^{CPL}$ does not discover a route between s and e, then $\mathbf{R}^{TD}$ does not discover a route. However, the converse does not hold. In particular,

$$\mathbb{E}\left(Succ^{\mathbf{R}^{CPL}}\right) \geq \mathbb{E}\left(Succ^{\mathbf{R}^{TD}}\right). \tag{8.18}$$

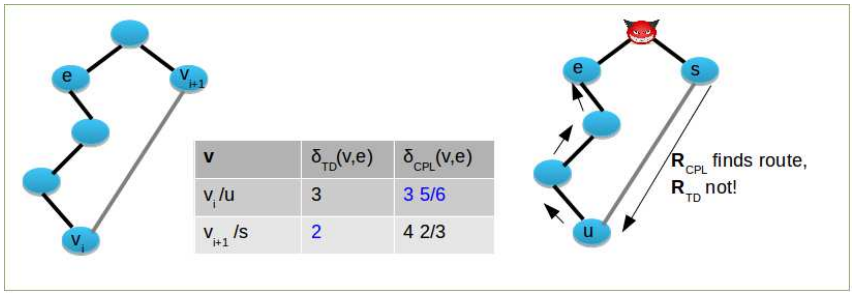


Figure 8.6: Illustrating the proof of Theorem 8.10: left: v_{i+1} is closer to destination e than v_i for distance δ_{TD} but not for δ_{CPL}; right: pair (s, e) for which $\mathbf{R}^{CPL}$ is successful as s forwards to u, but $\mathbf{R}^{TD}$ is not successful because s forwards to the attacker.

Proof. We prove the claim by contradiction. Assume that there is pair s, e such that the algorithm $\mathbf{R}^{TD}$ terminates successfully while $\mathbf{R}^{CPL}$ does not. Let $p = (v_0, v_1, \ldots, v_l)$ with $v_0 = s$ and $v_l = e$ denote the discovered route. By Lemma 8.9, p is a greedy path for distance δ_{TD} but not for δ_{CPL}. In other words, there exists $0 \leq i < l$ such that i) $\delta_{TD}(id(v_{i+1}), id(e)) < \delta_{TD}(id(v_i), id(e))$ and ii) $\delta_{CPL}(id(v_{i+1}), id(e)) \geq \delta_{CPL}(id(v_i), id(e))$. By the definitions of both distances in Equation 8.1 and Equation 8.2, this implies that $cpl\,(id(v_{i+1}), id(e)) < cpl\,(id(v_i), id(e))$ and $|id(v_{i+1})| < |id(v_i)|$. In other words, v_{i+1}'s coordinate has a lower common prefix length to $id(e)$ and is shorter than $id(v_i)$. The right side of Figure 8.6 displays an example.

We base our contradiction upon the following observation concerning routes in trees. Consider the *tree route* between two nodes, i.e., the path between them using only tree edges. Along the tree route, the common prefix length stays constant until the least common ancestor is reached and then increases. Now, if $id(v_{i+1})$ has a shorter common prefix with $id(e)$ than $id(v_i)$, v_{i+1} is not contained in the tree route. Furthermore, as the routing algorithm $\mathbf{R}^{CPL}$ does not successfully discover a route, the attacker has to control one node on the tree route.

We can use the above observation to establish contradictions for both strategies *ATT-RAND* and *ATT-ROOT*. Note that if the common prefix length decreases when forwarding to v_{i+1}, we need to have $cpl\,(id(v_i), id(e)) > 0$. For the attack strategy *ATT-RAND*, the attacker on the tree route is either an ancestor of v_i or of e. However, the attacker replaces the prefixes of all its children and hence descendants, so that the perceived common prefix length of v_i' and e's coordinates should be 0 unless there exists an attacker-free tree route. This is a clear contradiction. Similarly, if $cpl\,(id(v_i), id(e)) > 0$, v_i and e have

a common ancestor aside from the root. In particular, the tree route does not pass the root. When applying *ATT-ROOT*, the only attacker is the root, which again contradicts the existence of an attacker on the tree route. Thus, we have shown by contradiction that $\mathbf{R}^{TD}$ only succeeds if $\mathbf{R}^{CPL}$ does.

Thus, we have shown that indeed Equation 8.17 holds. Equation 8.18 is a direct consequence as it averages over all source-destination pairs and systems. It remains to show that the converse of Equation 8.17 does not hold. In other words, there exist instances when $\mathbf{R}^{CPL}$ terminates successfully while $\mathbf{R}^{TD}$ fails. We display such an example in Figure 8.6. □

While we can show that our modifications are indeed enhancements, our theoretical analysis does not provide any absolute bounds on the success ratio. In particular, we cannot compare the success ratio of our approach to that of virtual overlays by theoretical means only.

8.4.3 Simulations

We first describe our simulation model and set-up, extending the model from Section 8.3.2. Afterwards, we state our expectations and results. We close this section by a short discussion of the results.

Simulation Model and Evaluation Metrics: We extend the simulation model from Section 8.3.2 based on the methodology described in Section 5.4. Thus, we simulate the robustness of an overlay by subsequently selecting random failed nodes. In each step, we select a certain fraction of additional failed nodes and then determine the success ratio. Furthermore, we evaluate attacks using the two attack strategies *ATT-RAND* and *ATT-ROOT* described above. We first establish the overlay applying the respective attack strategy and then execute the routing for randomly selected source-destination pairs of responsive nodes.

We compare our results to the virtual overlay VO, described in Section 8.3. Our attacker on VO does not manipulate the tunnel establishment but merely drops requests. Recall that routing in VO relies on a Kademlia DHT such that neighbors in the DHT communicate via a tunnel of trusted links, as described in Section 3.3. The routing between two DHT neighbors thus fails if the attacker is contained in the tunnel. However, if routing between two overlay neighbors fails, the startpoint of the failed tunnel can attempt to select a different overlay neighbor as long as it has one neighbor closer to the destination. We further enhance the success ratio of VO by allowing backtracking in the DHT. In addition, we also allow for shortcuts, i.e., rather than following the tunnel to its endpoint, nodes on the path can change to a different tunnel with an endpoint closer to the destination. Thus, we maximize the chance of successful delivery in VO by backtracking and shortcuts in addition to the use of non-strategic attacker.

Set-up: We simulated the embedding and routing algorithms as parametrized in Section 8.3.

In order to evaluate the robustness, we removed up to 50% of the nodes in steps of 1%. During the process of removing nodes, individual nodes inevitably became disconnected from the giant component, so that routing between some pairs was no longer possible. For this reason, we only considered the results for source-destination pairs in the same component. Our results are presented for 1, 5, and 15 trees only.

The number of edges x controlled by the adversary A were chosen as $x = 2^i \times \lceil \log_2 n \rceil$ with $0 \leq i \leq 6$ and $\lceil \log_2 n \rceil = 16$. So, up to $1,024$ attacker edges were considered. In particular, $x = 1024 > \frac{\sqrt{n}}{\log n}$, a common asymptotic bound on the number of edges to honest nodes considered for Sybil detection schemes as described in Section 3.4. For quantifying the achieved improvement, we compared our approach to the resilience of the original PIE embedding and routing, i.e., 1 tree, δ_{TD}, and no backtracking.

For VO, we used a degree of parallelism of $\alpha = 1$. Since backtracking was applied, all values of $\alpha > 0$ resulted in the same success ratio, because regardless of the value of α, the routing succeeded if and only if a greedy path in the virtual overlay existed. Thus, restricting our evaluation to $\alpha = 1$ did not impact our results with regard to the success ratio.

We averaged the results over 20 runs with $10,000$ source-destination pairs each. Results are presented with 95% confidence intervals.

Expectations: We expect that the use of backtracking already increases the success ratio considerably for $\gamma = 1$. However, for large failure ratios or a large number of attacker edges, the single-connected nature of the tree should result in a low success ratio. By using multiple trees, we expect to further increase the success ratio until close to 100% of the paths correspond to a greedy path and hence a route in at least one embedding.

For the robustness to failures, the original distance function *TD* should result in a higher success ratio than *CPL* because of its shorter routes, as seen in Section 8.3.2, and thus lower probability to encounter a random failed node. However, by Theorem 8.10, *CPL* increases the success ratio in contrast to the original distance.

Our first attack strategy, *ATT-RAND*, should not have a strong impact as the fraction of controlled edges is low and the attacker usually does not have an important position in the trees. In contrast, we expect many requests to be routed via the root, so that at least for a low number of trees, *ATT-ROOT* should be an effective attack strategy.

In comparison, our attack on VO does not enable the attacker to obtain a position of strategic importance, so that the impact of the attack should be much less drastic than *ATT-ROOT*. However, communication between DHT neighbors relies on one tunnel whereas tree embeddings provide multiple routes. Thus, when using multiple diverse trees, we expect our approach to be similarly effective as VO, possibly even more effective.

Results: While the results verified our expectations with regard to the advantage of the distance *TD* for random failures and the advantage of *CPL* for attacks, the observed differences between the two distances were negligible, i.e., less than 0.1%. Hence, we present the results for *CPL* in the following.

We start by evaluating the robustness to random failures. The results, displayed in Figure 8.7, indicate that the use of multiple embeddings considerably increased the robustness. The success ratio for $\gamma = 1$ was low, decreasing in a linear fashion to less than 30% for a failure ratio of 50%. In contrast, for $\gamma = 15$, the success ratio exceeded 90%. Though the number of embeddings was the dominating factor, the tree constructing algorithm also strongly influenced the success ratio. For $\gamma > 1$, aiming to choose distinct parents improved the robustness to failures because of the higher number of distinct routes. For example, when routing in 5 parallel embeddings, the success ratio was above 80% for *DIV-RAND*. In contrast, *BFS* had a success ratio below 70%. In summary, the robustness to failures was extremely high for multiple embeddings, enabling a success ratio of more than 95% for up to 20% failed nodes. The robustness was further increased by using *DIV-RAND* or *DIV-DEP* rather than *BFS*, showing that even such relatively simple schemes can achieve a noticeable improvement.

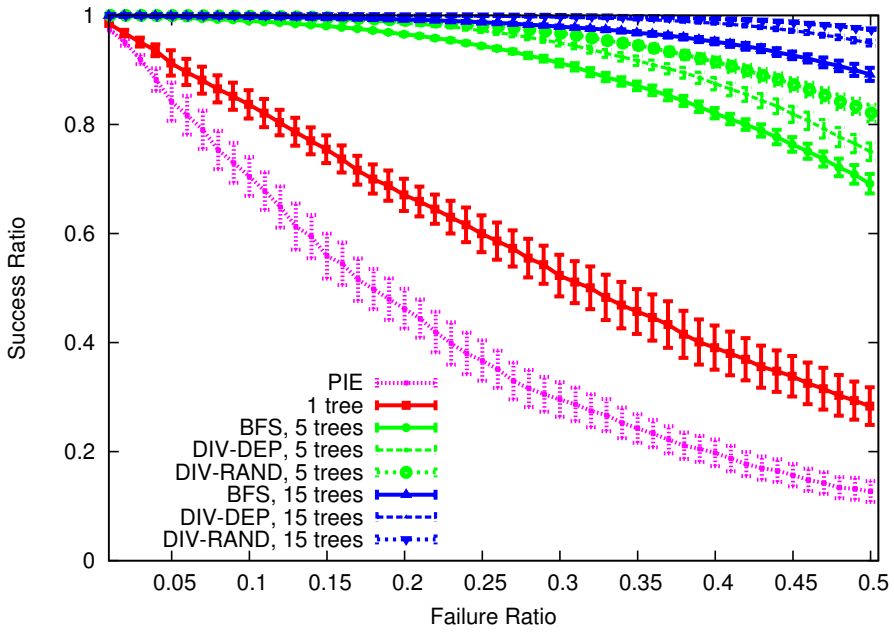
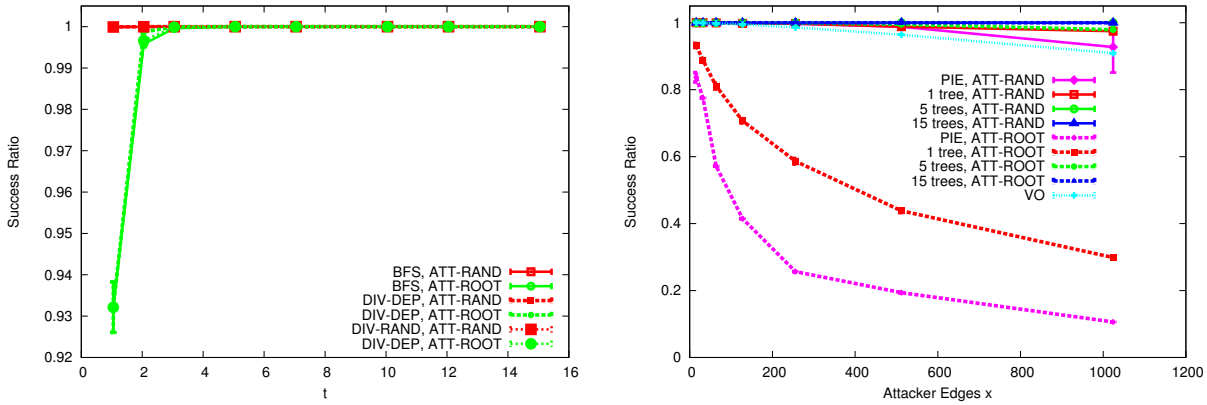


Figure 8.7: Robustness to random failures varying number of trees and tree construction

Now, we consider the censorship-resistance for $x = 16$ attacking edges, as displayed in Figure 8.8a. If the adversary A was unable to manipulate the root selection, the success ratio was only slightly below 100%. Even if $\gamma = 1$, more than 99.5% of the routes were successfully discovered. The high resistance against *ATT-RAND* was to be expected, considering that the attack was only slightly more severe than failure of one random node. If the attacker was able to become the root, the success ratio dropped to about 93% for $\gamma = 1$. However, with multiple trees, the ratio of *ATT-ROOT* was close to 100%. The impact of the tree construction was small but noticeable. So, *BFS* generally resulted in a slightly lower success ratio. Hence, by using multiple embeddings and backtracking, the resistance to an adversary that can establish only $\lceil \log_2 |V| \rceil = 16$ is such that nearly all routes are successfully found.

For an increased number of attacking edges x, the success ratio remained close to 100% when more than one tree was used for routing, as displayed in Figure 8.8b for *DIV-DEP*. However, for one tree, the success ratio decreased drastically if an attacker could undermine the root selection. For $x = 1024$, i.e., if the attacker controlled edges to roughly 1.7% of the nodes, the success ratio for $\gamma = 1$ decreased



(a) 16 Edges (b) Up to 1024 edges

Figure 8.8: Censorship-Resistance of tree routing for distance *CPL* to adversaries which are either able to undermine the root election (*ATT-ROOT*) or are unable to do so (*ATT-RAND*) for a) $x = 16$ attacking edges, and b) up to $1,024$ attacking edges and tree construction *DIV-DEP*

to slightly less than 30%. In contrast, if $\gamma = 5$ or $\gamma = 15$, the success ratio was still 97.9 or 99.9%, respectively. Figure 8.9 gives a more detailed analysis of the impact of the number of trees. For two trees, the success ratio nearly doubles to more than 60%. With 4 trees, the success ratio is above 95 %, outperforming *VO*.

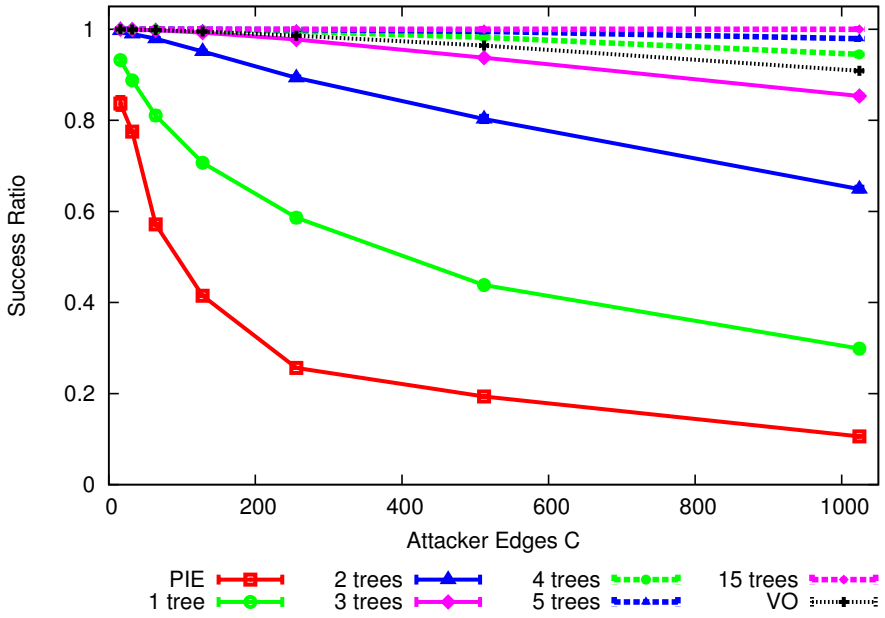


Figure 8.9: Resistance to *ATT-ROOT* for varying number of trees

In order to quantify the improvements provided by our resilience enhancements, we compared the results for our approach with the PIE embedding. As can be seen from Figure 8.7, the success ratio dropped much more quickly for PIE than for the improved approaches. For an adversary with $x = 16$ connections to honest nodes, PIE suffered from more than twice the numbers of failed requests than all other approaches (Figure 8.8b) because it relies on only one tree and does not apply backtracking. When increasing the number of attacker edges, the success ratio dropped further to less than 15% for $x = 1024$. Our approach achieved more than twice the success ratio even for $\gamma = 1$.

In contrast to PIE, VO exhibited a rather high success ratio as displayed in Figure 8.8b. VO's advantage in contrast to $\gamma = 1$ holds despite VO's longer routes (see Section 8.3.2). The reason for VO's lower vulnerability lies in the absence of strategic manipulation. While greedy embeddings allow the attacker to assume an important role, our attacker in VO does not attract a disproportional fraction of traffic. However, establishing multiple trees ensures that the role of the root is effectively mitigated, so that the censorship-resistance of VO is slightly lower than VOUTE's resistance for 5 or more parallel

embeddings.

Discussion: We have shown that multiple embedding and backtracking enable high resilience, outperforming state-of-the-art approaches. Here, we focused on node-to-node communication. While content retrieval results in longer routes, we expect the success ratio to be similar as backtracking in the DHT allows the use of multiple paths. In addition, the number of replicas per content can be adjusted to increase the success ratio.

While Theorem 8.10 shows the advantage of *CPL* in the presence of failures, the actual advantage is negligible, so that it seems more sensible to use the original distance *TD* due to its higher efficiency.

In summary, our enhancements to the robustness and censorship-resistance were both needed and highly effective.

8.5 Anonymity and Membership-Concealment

In this section, we evaluate the anonymity and membership-concealment of VOUTE. We use the methodology discussed in Section 5.5. First, we show that anonymous return addresses provide possible innocence. Second, we conjecture that the revealed topology information does not provide enough information about non-neighboring nodes to breach the membership-concealment.

8.5.1 Anonymity

We show possible innocence of both the sender and receiver. Note that the proof assumes that the attacker's view of the overlay is restricted. In particular, we assume that it cannot be sure to know all neighbor's neighbors. We complement the theoretical analysis with some empirical results. While estimating the actual degree of anonymity depends on various parameters, the essential result from Theorem 8.11 is that the attacker cannot differentiate if the request is addressed to a node or descendants of the node. In particular, nodes of degree 1 choose the same parent consistently, so that a high fraction of nodes with neighbors of degree 1 indicates a high uncertainty in identifying the receiver. Thus, we evaluate this property for our exemplary social networks.

Theorem 8.11. *Let u be a local attacker. Consider a request addressed to return addresses $y = (y_1, \ldots, y_\gamma)$ with routing information $\tilde{k} = (\tilde{k}_1, \ldots, \tilde{k}_\gamma)$ for γ embeddings generated by Algorithm 8.3. Let A be a polynomial-time algorithm executed by u to identify the sender s or receiver e of the request. Then u cannot identify $v \in \{s, e\}$ with absolute certainty, i.e., the anonymity of v is $ano_{\mathbf{Sys}}(v) > 0$ for all systems Sys consisting of a graph with γ modified PIE embeddings.*

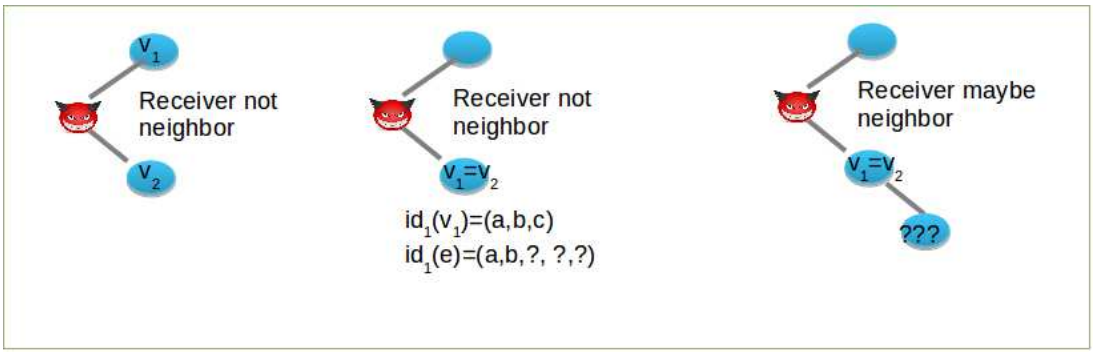


Figure 8.10: Illustrating the proof of Theorem 8.11: Let v_1 and v_2 be the neighbors with the closest coordinates to the receiver's coordinates $id_1(e)$ and $id_2(e)$ in the first and second embedding, respectively. The attacker can only infer if neighbors are not the receiver e but can not tell if they are. In particular, the attacker A knows the common prefix of the receiver's coordinates and the coordinates of its neighbors but not the remaining elements of the coordinate, as indicated by the ?s in the coordinate. In the first scenario, $v_1 \neq v_2$ shows that the receiver is not a neighbor. In the second scenario, A can infer that the third element of the receiver's coordinate $id_1(e)$ is not c and hence v_1 is not the receiver. In the last scenario, A is unable to tell if a neighbor is indeed the receiver or if a child of the neighbor is the receiver.

Proof. We start by considering receiver anonymity. We consider three cases and show for each case that either i) the attacker can determine that the receiver e is not a neighbor but cannot infer the coordinate of the actual receiver or ii) the attacker remains uncertain if the receiver is a neighbor or a neighbor's descendant. We illustrate the cases in Figure 8.10. Throughout the proof, let $v_i \in N_u$ be the closest neighbor of u to y_i for $i = 1, \ldots, \gamma$.

First, assume there exist i, j such that $v_i \neq v_j$. It follows that none of u's neighbors is the receiver due to the fact that the receiver can be identified as the closest node to all return addresses. So, u, not being aware of the remaining nodes and their coordinates in the system, cannot identify the receiver.

For the second case, assume that indeed $v_i = v_j$ for all $1 \leq i, j, \leq \gamma$ but there exists an i such that $cpl(hc(id(v_i), \tilde{k}_i), y_i) < |id(v_i)|$, i.e., the common prefix length of $id(v_i)$ and the target coordinate is less than the length of v_i's coordinate. Then v_i cannot be the receiver because at least the last element in the i-th coordinate of v_i does not agree with $id_i(e)$. So, again the receiver is not a neighbor of u and hence u is unable to identify e due to its limited view of the overlay.

Third, assume that indeed $v_i = v_j$ for all $1 \leq i, j, \leq \gamma$ and $cpl(hc(id(v_i), \tilde{k}_i), y_i) = |id(v_i)|$ for all i. Then, the node v_i can potentially be the receiver but so can any node w that is a descendant of v_i in all trees. Any return address vector of w would result in the same results as a return address vector of v_i from u's local point of view. Due to its restricted topology knowledge, u is unaware if such a descendant w exists, and hence might guess that $e = v_i$ but cannot be certain.

Thus, receiver anonymity follows as the return address does not allow the unique identification of the receiver. Sender anonymity follows analogously as a node can always forward a request from a child. □

Estimating Attacker Certainty: Theorem 8.11 has the limitation that the degree of anonymity remains unclear. However, as illustrated by the third scenario in Figure 8.10, an attacker cannot be sure if a neighbor is the receiver or a node that is a descendant of the neighbor in every tree at all times. Note that nodes of degree 1 have to choose the same parent in all trees and at all times. Thus, the probability that a node has at least one neighbor of degree 1 present a plausible lower bound on the attacker's uncertainty.

For this reason, we now determine the fraction f of nodes that have a neighbor with degree 1 for exemplary social networks. Note that f does not correspond to the fraction of nodes with degree 1 because one node can have multiple neighbors with degree 1. Formally, for a graph $G = (V, E)$, let

$$nd\colon V \to \{0, 1\}$$
$$nd(v) = \begin{cases} 1, \exists w \in N(v) : N(w) = \{v\} \\ 0, \text{ otherwise} \end{cases}.$$

Then, we define the fraction $f(G)$ as the average of ng, i.e.,

$$f(G) = \frac{1}{|V|} \sum_{v \in V} nd(v). \tag{8.19}$$

We have evaluated Equation 8.19 for our three social networks *FB*, *SPI*, and *WOT*. Indeed, the fraction of nodes with neighbors of degree 1 is high enough to result in a non-negligible uncertainty in determining the receiver. More precisely, the fraction of such nodes is 10.64%, 9.75%, and 12.41 % for *FB*, *SPI*, and *WOT*, respectively. While the presented numbers do not directly relate to the ratio of incorrect accusations, the results indicate that our return addresses indeed lead to a considerable uncertainty in identifying communicating parties.

Note that the above bounds only consider the case when the sender or receiver is a neighbor of the attacker. If a node does not establish a trust relationship with the attacker, a local attacker should be unaware of the node's membership in the F2F overlay and thus unable to link requests to the node.

We have shown that return addresses prevent the identification of the receiver. Note that the proof of Theorem 8.11 does not actually require the encryption of the coordinates by the application of a hash cascade. Instead, a simple padding can achieve the same uncertainty. However, unnecessary topology information is revealed without the encryption, as detailed in the following.

8.5.2 Membership-Concealment

Membership-concealment especially requires hiding the topology of the overlay, as it corresponds to a social graph and might allow the attacker to identify participants. As the overall aim of an attacker A is to identify a node in the network, A tries to gather as much information about a node as possible.

In the original embedding, A would gather all coordinates and derive information about the social graph from the parent-child relations. Furthermore, A could track the number of messages addressed to certain coordinates and the responses. In this manner, A could to a certain extent track the activity of users and possibly correlate the information with external sources.

The application of hash cascades prevents A from i) collecting the coordinates in the overlay, ii) correlating requests to the same node when using different addresses, and iii) determining parent-child relations as only the common prefix with A's neighbors is revealed. While we cannot prove that we provide membership-concealment, we argue that the revealed topology information is insufficient for reconstructing the social graph.

One attack strategy of the attacker would be to infer further elements of the original coordinate aside from the common prefix with neighbors. Note that the elements of the receiver address corresponds to hashes of pseudo-random numbers, so that inferring information about the coordinate is a contraction to the preimage-resistance of the hash function or the pseudo-randomness of the generated numbers. A formal proof is out of scope for this thesis. However, the idea of such a proof would be to transfer the concept of pre-image resistance to (padded) coordinates. Then, we could show that the chance of an attacker to determine the coordinate corresponding to a return address is negligible. The proof would rely on the assumptions that the hash function h is pseudo-random and preimage-resistant and the pseudo-random number generator is secure. For instance, a sequence of games proof [17] seems a suitable proof strategy, subsequently deriving the advantage of the attacker starting with a game corresponding to the actual scenario and ending at an attacker that has to find a preimage of h to any of L hashed values corresponding to the elements of a return address. However, an in-depth treatment of cryptographic primitives is out of scope for now. We rather focus on the consequences of hiding coordinates on the topology inference.

Now, we assume that the attacker can indeed not infer any information about receiver coordinates apart from the common prefix length with its neighbors. For simplicity, consider one embedding and assume that an attacker A gathers the return addresses contained in requests with the goal of reconstructing the overlay topology. As the return addresses only reveal the common prefix length of the receiver with each of A's neighbors, A can divide the receivers into classes based on these common prefix lengths. However, it cannot tell if two distinct addresses within one class belong to the same node or even neighboring nodes. Thus, A is unable to tell anything about the relation between return addresses apart from the fact that they have a similar relation to its neighbors. Note that if the coordinates were only padded, A could always infer the common prefix length of two coordinates. In the presence of return addresses, even that is impossible for nodes within the same class. In particular, A is unable to derive typical graph metrics such as the degree or the number of children within a spanning tree.

So, A's only information is the number of gathered return addresses per class. This is only of limited value because measurements indicate that the activity of users [134] and thus the number of return addresses they generate is heterogeneous. Thus, a large set of return addresses in one class does not necessarily indicate that the number of distinct users within the class is high. It seems highly unlikely that A can derive the identity of users in an F2F overlay from the available data.

In summary, our schemes does not only achieve possible innocence, we also reduce the available topology information to a degree that should provide effective membership-concealment.

8.6 Discussion

We have designed a novel F2F overlay based on network embeddings that satisfies all our requirements. In particular, we realized the five algorithms from Section 2.6.1 as follows:

- The routing algorithm $\mathbf{R}_{node}$ detects a path from a source node s to a destination e given return addresses of e by applying Algorithm 8.2 in τ parallel embeddings.
- The routing algorithm $\mathbf{R}_{content}$ uses DHT routing in a virtual overlay, levering $\mathbf{R}_{node}$ to find paths between virtual neighbors.
- The address generation algorithm $\mathbf{AdGen}_{node}$ generates anonymous return addresses of a destination e based on e's coordinates in the embeddings based on Algorithm 8.3.
- The address generation algorithm $\mathbf{AdGen}_{content}$ generates file keys by hashing the content of a file into the identifier space of the virtual overlay.
- The stabilization algorithm $\mathbf{S}$ locally repairs the embeddings according to Algorithm 8.1 and updates the routing tables entries in the DHT.

Property	Theory	Simulation	
Efficiency and Scalability			
Efficient Routing	$\mathcal{O}\left(\tau\frac{\gamma}{q}\log n\right)$	Shorter routes	✓
Balanced Content Addressing	$\mathcal{O}\left(\frac{\log n}{n}\right)$	As in SOTA	✓
Efficient Content Discovery	$\mathcal{O}\left(\tau\left(\frac{\gamma}{q}\right)\log^2 n\right)$	Barely longer routes	✓
Efficient Stabilization	$\mathcal{O}\left(\gamma\frac{\gamma}{q}\log n + c_{DHT}\right)$	Lower overhead	✓
Robustness and Censorship-Resistance			
Robustness	Improvements of standard GE	Improved	✓
Censorship-Resistance	Improvements of standard GE	Improved	✓
Anonymity and Membership-Concealment			
Sender	Possible Innocence	Graph-specific Bounds	✓
Receiver Anonymity	Possible Innocence	Graph-specific Bounds	✓
Membership-concealment	Improvements	(no eval)	✓?

Table 8.1: VOUTE's performance with regard to requirements using mathematical analysis and simulation in comparison to the state-of-the-art (SOTA), Parameters: γ embeddings of which τ are used for routing, q: probability to accept suboptimal parent, c_{DHT}: stabilization complexity virtual overlay; GE=greedy embedding

Table 8.1 summarizes the results of our evaluation with regard to the eight requirements introduced in Section 2.6. For the table, we consider the original tree distance δ_{TD} rather than our modified common prefix length-based distance function $dist_{CPL}$ as the latter did not improve the censorship-resistance considerably and increased the communication complexity.

With regard to efficiency and scalability, we formally proved the logarithmic or polylogarithmic bounds on the routing, content discovery, and stabilization complexity of the spanning trees. The stabilization of the virtual overlays and the balanced content addressing follow from the corresponding results for DHTs. Our simulation study confirms the reduced communication complexity in comparison to the state-of-the-art. Though our simulations indicated a slightly increased overhead for content discovery in comparison to virtual overlays, virtual overlays achieve this slight advantage at the price of an unacceptable stabilization complexity. Thus, we consider the slightly increased overhead for one measured quality acceptable, as, in contrast to the state-of-the-art approaches, we manage to achieve all requirements with regard to scalability and efficiency.

Meaningful theoretical bounds on the robustness and censorship-resistance of a F2F overlay require hard assumptions about the graph structure, such as the existence of independent paths between nodes. In order to avoid such assumptions, we evaluated the robustness and censorship-resistance mainly based on simulations using real-world data rather than theoretical means. Our theoretical bounds only show that we enhanced the robustness and censorship-resilience of greedy embedding in contrast to the standard greedy embedding but do not offer a comparison to other state-of-the-art approaches. In our simulation study, the observed resilience to failures and attackers was very high. Up to 30 % of failed nodes could be tolerated without reducing the success ratio below 95 %. Two types of attacks were considered: When an adversary was unable to manipulate the root election protocol, even adversaries establishing connections to more than a thousand nodes could be tolerated without a significant drop in the success ratio. In contrast, if an attacker controlled the root of all spanning trees, the attack reduced the success ratio more drastically. Yet, these attacks could also be counteracted to a large degree when a high degree of parallelism was applied. In summary, VOUTE considerably increases the robustness and censorship-resistance of greedy embeddings and indeed achieves a higher resilience than virtual overlays. VOUTE can thus be applied in dynamic, possibly malicious environments.

Providing sender and receiver anonymity is highly important for privacy-preserving communication, even if the communication is routed via friends. In our evaluation, we showed that possible innocence with regard to sender anonymity holds by default as long as the adversary does not observe the complete neighborhood of a sender. In order to achieve receiver anonymity, we obfuscated receiver's coordinate in the embedding such that the receiver cannot be uniquely determined based on the obfuscated address. Again, we proved receiver anonymity under the assumption that the adversary does not observe the complete neighborhood of a receiver. We complement our proof of possible innocence with graph-specific upper bound on the certainty of the attacker in identifying the receiver or sender, which indicates a non-trivial uncertainty of the attacker even if it is the first or last node on the route. Furthermore, our algorithms reveal the considerably less information about the receiver's coordinate without increasing the communication complexity. However, we did not explicitly show that we achieve membership-concealment towards untrusted participants. By obfuscating the receiver addresses, the topology of the social graph

cannot be easily reconstructed from the anonymous addresses. In this manner, we are likely to prevent the identification of users due to structural similarities to publicly known social networks. As nodes do not establish direct connections with the devices of untrusted participants and are not required to reveal any identifying information apart from their anonymous address, identifying participants and thus breaking the membership-concealment seems an unfeasible task. Yet, we are unable to provide any guarantees with regard to membership-concealment, thus the question mark in Table 8.1. Nevertheless, we considerably improved the privacy provided in greedy embeddings. In contrast to deployed systems such as Freenet, we provide guarantees on the anonymity.

Despite fulfilling all requirements, there remain some concerns with regard to a real-world implementation. For instance, while we can guarantee that a local adversary is unable to identify the receiver uniquely, it might identify the receiver with a high probability. For example, if a neighbor is the closest neighbor to the destination in all γ parallel embeddings, the adversary might safely assume that the neighbor is the receiver and persecute said neighbor. In the future, we plan to consider an increased anonymity at the price of lower efficiency.

A complementary concern is related to the construction and stabilization of spanning trees in dynamic scenarios. More precisely, the effectiveness of distributed spanning trees algorithms such as [120] depends highly on the scenario. Keeping the time required to build a tree low is essential for a smoothly working system. Depending on the frequency of topology changes, nodes have to adapt their frequency of checking the availability of parent nodes. In addition, potential speed-ups such as the preemptive selection of alternative parents should be evaluated. Therefore, building a real-world realization of the proposed system requires considerable effort for finding fast concrete stabilization mechanisms.

In Chapter 9, we discuss the implementation of a real-world prototype of VOUTE and additional, more general directions of future work.

Chapter 9

Conclusion

In this thesis, we enhanced the state-of-the-art with regard to F2F overlays through

1. Development of a common methodology and acquisition of suitable data sets based on measurements in Freenet,

2. Evaluation of existing approaches with the identification of concrete reasons for their deficits, and

3. Design and Evaluation of VOUTE, a F2F overlay based on network embeddings.

In particular, we realized that virtual overlays require an underlying routing schemes for efficient messaging and content addressing. Thus, we showed how network embeddings can be adapted to provide such an underlying routing scheme while meeting the security and privacy constraints of F2F overlays. Both theoretical and simulation-based evaluation indicate the advantage of our design in contrast to the state-of-the-art in terms of efficiency and resilience to failures and attacks. We thus achieved significant progress in the area of F2F overlays, as emphasized by numerous publications [82, 130, 131, 132, 134, 135, 136, 137, 138, 143].

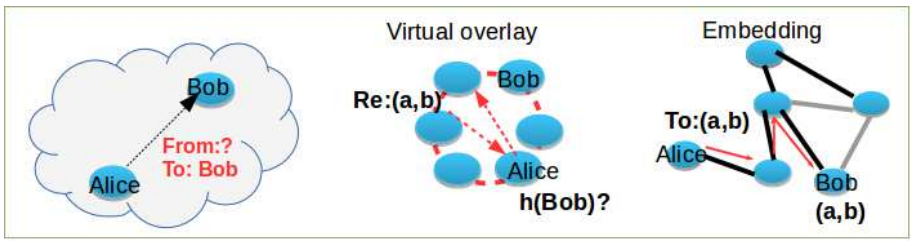
In Chapter 1, we derived our desired basic functionalities, namely messaging and content sharing, based on three use cases. Throughout the thesis, we then focused on realizing these functionalities in a very general manner without directly considering the initial use cases. Now, we close the circle by specifying how we can realize the use cases in the context of VOUTE.

1. Our first scenario, displayed in Figure 9.1a, considers the whistle blower Alice aiming to contact the journalist Bob who has published his contact information in the F2F overlay. In the context of VOUTE, the published contact information can be in form of embedding coordinates. However, the overlay topology and hence the coordinates change constantly, so that Bob is required to explicitly contact each potential communication partner and update all sources containing his contact information whenever his coordinate(s) changes. Alternatively, Bob can publish a key pointing to an entry in the virtual overlay storing his coordinate(s) in order to only update one entry in the virtual overlay. Alice can then first retrieve the coordinate(s) using a virtual overlay lookup and then contact Bob using the underlying routing scheme of the network embeddings, as shown on the right of Figure 9.1a. Subsequent messages only require routing in the embedding, at least in the absence of further coordinate changes.

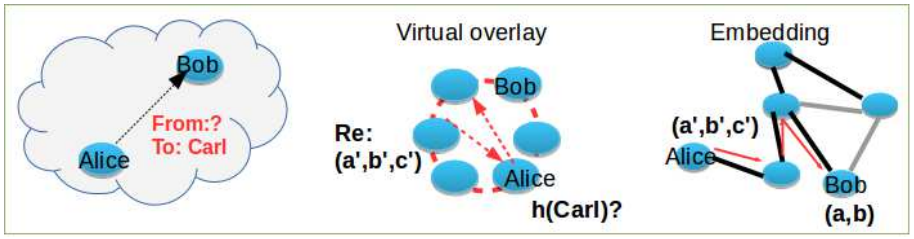
2. In our second scenario, Alice wants to contact Bob anonymously to ask a question with regard to a forum entry about some health issue. The difference to our first scenario is that Bob wishes to remain anonymous. For this reason, he stores his return addresses rather than his actual coordinates in the virtual overlay, as shown in Figure 9.1b. Otherwise, the communication process is identical to the first scenario. However, Bob might store multiple return addresses under different keys in order to prevent linking entries in different forum threads to the same identity.

3. Our third scenario has Bob publishing sensitive information that Alice wants to retrieve. Both publication and retrieval of content are realized by a virtual overlay lookup terminating at the closest node to the key associated with the information, as shown in Figure 9.1c.

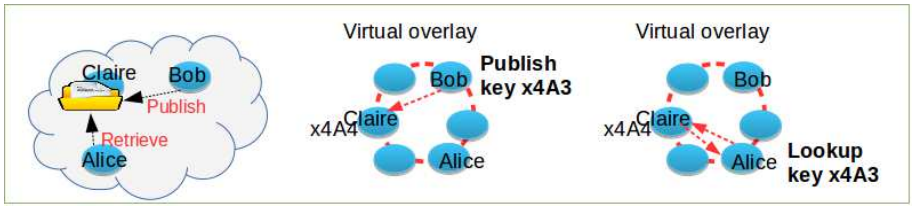
We can indeed realize our initial use cases such that the realization is i) *efficient and scalable* in terms of the communication complexity, ii) *robust and attack-resistant* in order to provide the required service despite a dynamic user group and potential denial-of-service attacks, and iii) *anonymous and membership-concealing.*



(a)



(b)



(c)

Figure 9.1: Exemplary use cases for F2F overlays and their realization in VOUTE (see Figure 1.2 for comparison): a) messaging with anonymous sender: Sender retrieves coordinate(s) in virtual overlay and sends message using the underlying routing scheme of the embedding, b) messaging with two anonymous parties using a pseudonym of the receiver: sender looks up pseudonym and retrieves anonymous address for routing rather than coordinate, c) anonymous content sharing storing the published content at an otherwise uninvolved user: Users perform a virtual overlay lookup for the content's key either for publication at the responsible node or retrieval of the content.
dotted black arrows indicate abstract functionality as perceived by the user, dotted red arrows virtual overlay lookup, solid red arrows routing in the embedding

While the concrete realization of more complex applications on the basis of our fundamental work is certainly a direction of future work, we identified additional future research questions. These research questions concern both i) the further enhancement and implementation of F2F overlays, and ii) the transfer of methods and results to other areas of research. In the following, we shortly introduce the identified areas of future research.

Dynamic Simulation and Prototype Implementation: As this thesis addressed the question of the inherent suitability of F2F overlay approaches rather than the evaluation of concrete approaches, we did not specify all aspects of the VOUTE architecture. In particular, we did not specify and evaluate protocols that are dependent on scenario-specific parameters such as node-to-node latencies and bandwidths. Our abstract methodology is currently unable to consider such aspects and thus needs to be extended. In order to evaluate the impact of delays and resource restrictions on the system performance, packet-level simulation environments such as OMNeT++ [14] should be utilized, followed by a real-world prototype and a testbed evaluation. Integrating our design into the Freenet client is a promising option, as it allows leveraging the broad user base of Freenet and only requires changing the routing and embedding protocol of the current Freenet implementation. We discussed several of the concrete challenges related to a prototype implementation in Chapter 8, in particular the choice and parametrization of the spanning tree construction algorithm.

Attack Detection and Consideration of Additional Attacks: Up to now, we only considered attack-resistance but not attack detection. Though our design exhibited a high resistance to attacks, detecting attackers and excluding them from the communication can further improve the quality-of-service and possibly deal with stronger attackers. Attack detection recognizes suspicious behavior and relies upon

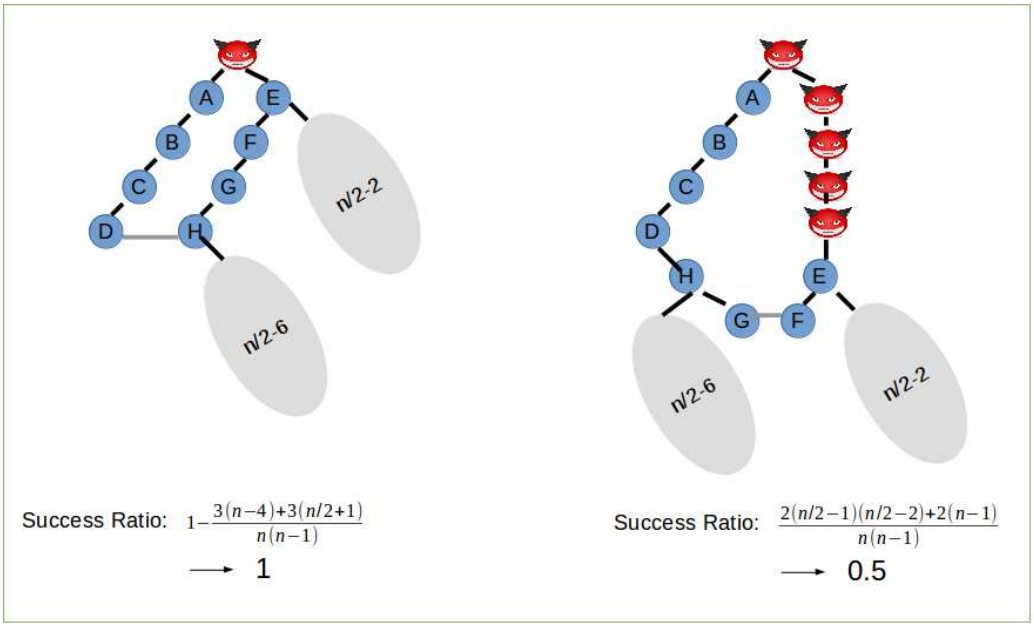


Figure 9.2: Example that only manipulating root election is not necessarily the best attack strategy ($\gamma = 1$ embedding, spanning trees constructed by a breadth-first search).
left: The attacker is the root dropping all requests routed via it. Then only the 3 nodes A, B, and C cannot route to the right subtree, while E, F, G and the subtree rooted at E cannot route to the left subtree, which consists of 4 nodes only. The success ratio is close to 1 for large overlays.
right: The attacker controls the root and adds four descendants. Here, only G and F can route to the other subtree. As each subtree is of size $n/2$, the success ratio converges to $1/2$.

global or local reputation schemes to judge the reliability of nodes. Purely local reputation schemes such as [106, 167] avoid the vulnerability of global reputation schemes, which spread the knowledge about potential malicious nodes throughout the system, to false accusations. In order to locally rate the trustworthiness of neighbors, nodes assign a trust level to each neighbor and adjust the level based on the ratio of requests forwarded to the neighbor that have received a positive reply. Due to the local nature of F2F overlays and the inability of nodes to simply change their neighbors if their current neighbors have lowered their rating, local attack detection schemes seem promising for F2F overlays and should be evaluated in the future.

Furthermore, we only focused on denial-of-service attacks through dropping of messages. Indeed, as motivated in Section 2.4, a modified Black Hole attack, i.e., an attacker manipulating the embedding such that the majority of requests is routed via malicious nodes who drop all requests, is a prevalent concern for structured approaches such as network embeddings. However, polluting content or overloading the overlay with fake requests are also pressing concerns, as are localized Eclipse attacks, which aim to censor specific keys in a DHT. Future work has to evaluate if the existing protection schemes such as [46, 95, 96] are indeed applicable in VOUTE.

Last, our attack model is restricted to a local adversary. However, without the use of effective steganography, powerful parties such as colluding ISP providers might identify nodes participating in the overlay. Such an attacker can then insert local adversaries into the overlay in a strategic manner, i.e., by befriending specific nodes or manipulating the tree structure strategically rather than only taking over the root of the spanning tree. For instance, Figure 9.2 illustrates that only manipulating the root election might not be the optimal strategy if the attacker knows the topology. Thus, future work should answer the question to which extent a global passive attacker in combination with a local attacker establishing x edges to honest nodes can strategically manipulate the embeddings in order to maximize the failure ratio.

Furthermore, the current scheme does not provide anonymization towards such a global passive adversary, as the identity of communicating parties can be revealed through traffic analysis. We now comment on the topic of anonymization in greater detail.

Anonymity-Efficiency Trade-off: The current design of VOUTE provides anonymity against a local adversary by replacing identifying coordinates with return addresses. However, the level of anonymity is limited to possible innocence if the adversary is a neighbor of the sender or receiver. The similarity of the anonymous return address and a neighbor's coordinate allows an attacker to identify possible receivers of

a message though it prevents definite identification. If such a suspicion is sufficient cause for persecution or in the presence of a global adversary, the achieved level of anonymity needs to be improved.

It seems possible to integrate known anonymization methods such as MIX cascades [43] or onion routing [125] into F2F overlays. Such techniques can be integrated by adapting Pisces [111], as described in Section 3.4, to F2F overlays in order to select onion routers or mixes through random walks. However, while our current scheme does not increase the communication complexity, these schemes inevitable require longer routes and thus higher communication delays and overheads. In addition, they potentially reduce the resilience to failures and attacks because of the increased probability of encountering failed or malicious nodes on longer routes.

Balancing anonymity and efficiency (and resilience) is a common issue in many anonymization services [29, 71]. In particular, the question of finding optimal trade-offs in the sense that one quantity is maximal while satisfying certain bounds on the other is of general interest. Rather than only focusing on the concrete scenario of F2F overlays, future work should take more general settings into account when aiming to design suitable trade-offs for F2F overlays.

Transfer of Results to Related Areas: F2F overlays are connectivity-restricted overlays, meaning that nodes cannot directly communicate with any other node in the overlay. Thus, a routing scheme has to be established without changing the overlay connections. We utilized network embeddings for providing such a routing schemes, which have previously been suggested for other connectivity-restricted networks such as wireless sensor networks or as an alternative for IP routing in content-centric networking [56, 80, 87]. As a consequence, our results on network embeddings are of interest for the area of content-centric networking as well.

In particular, content addressing has not been adequately addressed despite its importance for content-centric networking. While Kleinberg showed that his hyperbolic embedding guarantees that routing for an arbitrary key always terminates at the node with the closest coordinate to the key, he did not consider how many keys are mapped to each node [87]. Our results in Chapter 7 indicate that the number of keys per node is highly diverse, overloading individual nodes. As a consequence, alternative content addressing schemes have to be considered in these areas as well.

In contrast to F2F overlays, content-centric networking within one autonomous system is not concerned with resilience to attacks, hiding the network topology, and exhaustive stabilization complexity. These three requirements of F2F overlays prohibit the use of the tree embeddings for balanced content addressing as that would require reliable information about the overlay topology. Hence, because the topology can be utilized to achieve balanced content addressing in the embedding itself, an additional virtual overlay is unnecessary for content-centric networking. Rather, the embedding can be constructed such that the content is assigned in a balanced manner. We contributed one approach for content addressing in tree-based embeddings, described in Appendix D. However, we require each node to know the complete graph and change the keys of the content whenever the topology changes. Thus, future work in the area of network embeddings should consider the question of how to reduce the required local state and allow for deterministic topology-independent key assignment for content.

Apart from the very specific question of content addressing, our methodology for the evaluation of routing and embedding algorithms is mainly independent of F2F overlays. Hence, it is of interest for distributed routing algorithms in general, in particular for dynamic systems, as already indicated by publications applying these methods in related fields [27, 72, 85, 116, 133, 139, 140, 141].

In summary, this thesis provides the foundation for a usable privacy-preserving communication system in order to protect freedom of speech. We have identified vulnerabilities of the existing approaches, offered immediate solutions for an increased security in the Freenet system, and developed a promising conceptual approach. In the process, we discovered future research questions and introduced novel widely applicable methods for the evaluation of distributed systems. Our theory-based evaluation showed that our approach offers the required functionalities and opens the door for the development of a real-world implementation within a popular privacy-preserving communication system.

Bibliography

[1] BitTorrent. https://joindiaspora.com/. Accessed: 2015-10-05.

[2] Diaspora. https://joindiaspora.com/. Accessed: 2015-10-05.

[3] eMule. http://www.emule-project.net/home/perl/general.cgi?l=1. Accessed: 2015-10-15.

[4] Facebook Data Loss. http://www.reuters.com/article/2013/06/21/us-facebook-security-idUSBRE95K18Y20130621. Accessed: 2015-10-05.

[5] Facebook Friendship Graph. http://konect.uni-koblenz.de/networks/facebook-wosn-links. Accessed: 2012-6-1.

[6] Freenet Applications. http://freesocial.draketo.de/. Accessed: 2015-10-02.

[7] Freenet Index Sites. https://wiki.freenetproject.org/Using_Freenet#Create_.26_publish_a_Freesite. Accessed: 2015-12-29.

[8] Freenet Wiki: FNPRHProbeRequest. https://wiki.freenetproject.org/index.php?title=FCPv2/ProbeRequest. Accessed: 2015-10-16.

[9] How Burglars Use facebook. http://www.ibtimes.com/how-burglars-use-facebook-target-vacationing-homeowners-1341325. Accessed: 2015-10-05.

[10] Iplane Latencies. http://iplane.cs.washington.edu/data/data.html. Accessed: 2014-08-12.

[11] Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks, author=Yao, Zhongmei and Leonard, Derek and Wang, Xiaoming and Loguinov, Dmitri, booktitle=ICNP, year=2006.

[12] National Security Letters. https://www.eff.org/de/issues/national-security-letters. Accessed: 2015-12-18.

[13] NSA Files Timeline. http://www.theguardian.com/world/2013/jun/23/edward-snowden-nsa-files-timeline. Accessed: 2015-10-05.

[14] OMNeT++. https://omnetpp.org/. Accessed: 2016-01-30.

[15] Privacy-preserving P2P Data Sharing with ONESWARM, author=Isdal, Tomas and Piatek, Michael and Krishnamurthy, Arvind and Anderson, Thomas, booktitle=ACM SIGCOMM Computer Communication Review, volume=40, number=4, year=2010.

[16] R: Least Square Error Fitting. http://stat.ethz.ch/R-manual/R-patched/library/stats/html/nls.html. Accessed: 2014-04-16.

[17] Sequences of Games: A Tool for Taming Complexity in Security Proofs, author=Shoup, Victor, journal=IACR Cryptology ePrint Archive, year=2004.

[18] Succinct Greedy Graph Drawing in the Hyperbolic Plane, author=Eppstein, David and Goodrich, Michael T, booktitle=Graph Drawing, year=2009.

[19] Supposed User Count OneSwarm. http://forum.oneswarm.org/topic/1465. Accessed: 2015-10-19.

[20] Top 10 Traffic Hogs. http://www.statista.com/chart/1620/top-10-traffic-hogs/. Accessed: 2015-10-05.

[21] Tor Traffic Confirmation Attack. https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack. Accessed: 2015-10-05.

[22] Vorratsdatenspeicherung. http://www.spiegel.de/netzwelt/netzpolitik/vorratsdatenspeicherung-die-wichtigsten-texte-zum-comeback-der-vds-a-1068480.html. Accessed: 2015-12-18.

[23] We Live in Public. http://www.theguardian.com/film/2009/nov/04/josh-harris-we-live-public. Accessed: 2015-10-05.

[24] Web-of-Trust Graphs. http://www.lysator.liu.se/~jc/wotsap/wots2/. Accessed: 2012-6-1.

[25] William Acosta and Surendar Chandra. Trace Driven Analysis of the Long Term Evolution of Gnutella Peer-to-Peer Traffic. In *Passive and Active Network Measurement*. 2007.

[26] Sudhanshu Aggarwal and Shay Kutten. Time Optimal Self-stabilizing Spanning Tree Algorithms. In *Foundations of Software Technology and Theoretical Computer Science*, 1993.

[27] Frederik Armknecht, Manuel Hauptmann, Stefanie Roos, and Thorsten Strufe. An Additional Protection Layer for Confidential OSNs Posts. In *ICC*, 2014.

[28] Abdalkarim Awad, Reinhard German, and Falko Dressler. Exploiting Virtual Coordinates for Improved Routing Performance in Sensor Networks. *IEEE Transactions on Mobile Computing*, 10(9), 2011.

[29] Adam Back, Ulf Möller, and Anton Stiglic. Traffic Analysis Attacks and Trade-offs in Anonymity Providing Systems. In *Information Hiding*, 2001.

[30] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esmaeil Mohammadi. ANOA: A Framework for Analyzing Anonymous Communication Protocols. In *CSF*, 2013.

[31] Salman A Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. *arXiv preprint cs/0412017*, 2004.

[32] Heinz Bauer. *Measure and Integration Theory*. Walter de Gruyter, 2001.

[33] Krista Bennett, Christian Grothoff, Tzvetan Horozov, Ioana Patrascu, and Tiberiu Stef. Gnunet-A Truly Anonymous Networking Infrastructure. In *PETS*, 2002.

[34] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In *Designing Privacy Enhancing Technologies*, 2001.

[35] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your Contacts are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *WWW*, 2009.

[36] Mario Bonk and Oded Schramm. Embeddings of Gromov Hyperbolic Spaces. In *Selected Works of Oded Schramm*. 2011.

[37] Michael Brinkmeier, Günter Schäfer, and Thorsten Strufe. Optimally DOS Resistant P2P Topologies for Live Multimedia Streaming. *TPDS*, 20(6), 2009.

[38] Johannes Buchmann. *Introduction to Cryptography*. Springer Science & Business Media, 2013.

[39] Fabian Bustamante and Yi Qiao. Friendships that Last: Peer Lifespan and its Role in P2P Protocols. *Web Content Caching and Distribution*, 2004.

[40] Matthew Caesar, Miguel Castro, Edmund B Nightingale, Greg O'Shea, and Antony Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs. *ACM SIGCOMM Computer Communication Review*, 36(4), 2006.

[41] Guido Caldarelli. Scale-free Networks: Complex Webs in Nature and Technology. *OUP Catalogue*, 2007.

[42] Bengt Carlsson and Rune Gustavsson. The Rise and Fall of Napster-An Evolutionary Approach. In *Active Media Technology*. 2001.

[43] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2), 1981.

[44] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of cryptology*, 1(1), 1988.

[45] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-hastings Algorithm. *The American Statistician*, 1995.

[46] Thibault Cholez, Isabelle Chrisment, Olivier Festor, and Guillaume Doyen. Detection and Mitigation of Localized Attacks in a Widely Deployed P2P Network. *Peer-to-Peer Networking and Applications*, 6(2), 2013.

[47] Ian Clarke, Scott G Miller, Theodore W Hong, Oskar Sandberg, and Brandon Wiley. Protecting Free Expression Online with Freenet. *Internet Computing, IEEE*, 6(1), 2002.

[48] Ian Clarke, Oskar Sandberg, Matthew Toseland, and Vilhelm Verendel. Private Communication through a Network of Trusted Connections: The Dark Freenet. *Network*, 2010.

[49] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies*, 2001.

[50] Sebastian Clauß and Stefan Schiffner. Structuring Anonymity Metrics. In *Workshop on Digital identity management*, 2006.

[51] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the Internet to Random Breakdowns. *Physical review letters*, 85(21), 2000.

[52] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Breakdown of the Internet under Intentional Attack. *Physical review letters*, 86(16), 2001.

[53] Tyson Condie, Varun Kacholia, Sriram Sank, Joseph M Hellerstein, and Petros Maniatis. Induced Churn as Shelter from Routing-Table Poisoning. In *NDSS*, 2006.

[54] Bernd Conrad and Fatemeh Shirazi. A Survey on Tor and I2P. In *ICIMP*, 2014.

[55] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A Privacy-preserving Online Social Network Leveraging on Real-life Trust. *IEEE Communications Magazine*, 47(12), 2009.

[56] Andrej Cvetkovski and Mark Crovella. Hyperbolic Embedding and Routing for Dynamic Graphs. In *INFOCOM*, 2009.

[57] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Security and Privacy*, 2003.

[58] George Danezis and Prateek Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

[59] Matteo Dell'Amico. Mapping Small Worlds. In *P2P*, 2007.

[60] Jochen Dinger and Hannes Hartenstein. Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-registration. In *ARES*, 2006.

[61] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-generation Onion Router. Technical report, DTIC Document, 2004.

[62] Wolfgang Effelsberg, Ralf Steinmetz, and Thorsten Strufe. *Benchmarking Peer-to-Peer Systems: Understanding Quality of Service in Large-Scale Distributed Systems*. Springer, 2013.

[63] Nathan Evans, Bartlomiej Polot, and Christian Grothoff. Efficient and Secure Decentralized Network Size Estimation. In *NETWORKING*. 2012.

[64] Nathan S Evans, Chris GauthierDickey, and Christian Grothoff. Routing in the Dark: Pitch Black. In *ACSAC*, 2007.

[65] Nathan S Evans and Christian Grothoff. R5N: Randomized Recursive Routing for Restricted-route Networks. In *NSS*, 2011.

[66] Pierre Fraigniaud and George Giakkoupis. The Effect of Power-law Degrees on the Navigability of Small Worlds. In *PODC*, 2009.

[67] Pierre Fraigniaud and George Giakkoupis. On the Searchability of Small-world Networks with Arbitrary Underlying Structure. In *STOC*, 2010.

[68] Michael J Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *CCS*, 2002.

[69] Dominik Gall, Riko Jacob, Andrea Richa, Christian Scheideler, Stefan Schmid, and Hanjo Täubig. Time Complexity of Distributed Topological Self-stabilization: The Case of Graph Linearization. In *LATIN*, 2010.

[70] Michael R Garey and David S Johnson. *Computers and Intractability*. wh freeman, 2002.

[71] Nethanel Gelernter and Amir Herzberg. On the Limits of Provable Anonymity. In *WPES*, 2013.

[72] Daniel Germanus, Stefanie Roos, Thorsten Strufe, and Neeraj Suri. Mitigating Eclipse Attacks in Peer-To-Peer Networks. In *IEEE CNS*, 2014.

[73] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load Balancing in Dynamic Structured P2P Systems. In *INFOCOM*, 2004.

[74] P. Krishna Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-sharing Workload. In *SOSP*, 2003.

[75] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. A Measurement Study of Napster and Gnutella as Examples of Peer-to-Peer File Sharing Systems, journal = Computer Communication Review, year = 2002.

[76] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, Analysis, and Modeling of BitTorrent-like Systems. In *IMC*, 2005.

[77] Kandi Haribabu, Dushyant Arora, Bhavik Kothari, and Chittaranjan Hota. Detecting Sybils in Peer-to-Peer Overlays using Neural Networks and CAPTCHAs. In *CICN*, 2010.

[78] Bernhard Heep. R/Kademlia: Recursive and Topology-aware Overlay Routing. In *ATNAC*, 2010.

[79] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. A Measurement Study of a Large-scale P2P IPTV System. *Transactions on Multimedia*, 9(8), 2007.

[80] Julien Herzen, Cedric Westphal, and Patrick Thiran. Scalable Routing Easy as Pie: A Practical Isometric Embedding Protocol. In *ICNP*, 2011.

[81] Russell A Hill and Robin IM Dunbar. Social Network Size in Humans. *Human nature*, 14(1), 2003.

[82] Andreas Hoefer, Stefanie Roos, and Thorsten Strufe. Greedy Embedding, Routing and Content Addressing for Darknets. In *NetSys*, 2013.

[83] Riko Jacob, Andrea Richa, Christian Scheideler, Stefan Schmid, and Hanjo Täubig. A Distributed Polylogarithmic Time Algorithm for Self-stabilizing Skip Graphs. In *PODC*, 2009.

[84] Riko Jacob, Stephan Ritscher, Christian Scheideler, and Stefan Schmid. A Self-stabilizing and Local Delaunay Graph Construction. In *Algorithms and Computation*, 2009.

[85] Shankar Karuppayah, Stefanie Roos, Christian Rossow, Max Mühlhäuser, and Mathias Fischer. ZeusMilker: Circumventing The P2P Zeus Neighbor List Restriction Mechanism. In *ICDCS*, 2015.

[86] Jon Kleinberg. The Small-world Phenomenon: An Algorithmic Perspective. In *STOC*, 2000.

[87] Robert Kleinberg. Geographic Routing using Hyperbolic Space. In *INFOCOM*, 2007.

[88] Sebastian Kniesburges, Andreas Koutsopoulos, and Christian Scheideler. Re-chord: A Self-stabilizing Chord Overlay Network. In *SPAA*, 2011.

[89] Yehuda Koren. On Spectral Graph Drawing. In *Computing and Combinatorics*. 2003.

[90] Marek Kuczma. *An Introduction to the Theory of Functional Equations and Inequalities: Cauchy's Equation and Jensen's Inequality*. Springer Science & Business Media, 2009.

[91] Emmanuelle Lebhar and Nicolas Schabanel. Almost Optimal Decentralized Routing in Long-Range Contact Networks. In *ICALP*, 2004.

[92] Peter M Lee. *Bayesian Statistics: An Introduction*. John Wiley & Sons, 2012.

[93] Derek Leonard, Vivek Rai, and Dmitri Loguinov. On lifetime-based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks. In *ACM SIGMETRICS*, 2005.

[94] Chris Lesniewski-Lass and M Frans Kaashoek. Whanau: A Sybil-proof Distributed Hash Table. NSDI, 2010.

[95] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W Ross. Pollution in P2P File Sharing Systems. In *INFOCOM*, 2005.

[96] Jian Liang, Naoum Naoumov, and Keith W Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *INFOCOM*, 2006.

[97] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *PODC*, 2002.

[98] Dahlia Malkhi, Florin Oprea, and Lidong Zhou. Omega Meets Paxos: Leader Election and Stability without Eventual Timely Links. In *Distributed Computing*. 2005.

[99] Sandeep Mane, Sandeep Mopuru, Kriti Mehra, and Jaideep Srivastava. Network Size Estimation in a Peer-to-Peer Network. *University of Minnesota, MN, Tech. Rep*, 2005.

[100] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy Neighbor's Neighbor: The Power of Lookahead in Randomized P2P Networks. In *STOC*, 2004.

[101] Chip Martel and Van Nguyen. Analyzing Kleinberg's (and other) Small-world Models. In *PODC*, 2004.

[102] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, 50(4), 2006.

[103] Jiří Matoušek. Bi-Lipschitz Embeddings into Low-dimensional Euclidean Spaces. *Commentationes Mathematicae Universitatis Carolinae*, 31(3), 1990.

[104] Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-Peer Information System based on the XOR Metric. In *Peer-to-Peer Systems*. 2002.

[105] Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. Scalable Onion Routing with Torsk. In *CCS*, 2009.

[106] Pietro Michiardi and Refik Molva. Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Advanced Communications and Multimedia Security*. 2002.

[107] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S Wallach. AP3: Cooperative, Decentralized Anonymous Communication. In *ACM SIGOPS European workshop*, 2004.

[108] Bivas Mitra, Fernando Peruani, Sujoy Ghose, and Niloy Ganguly. Analyzing the Vulnerability of Superpeer Networks against Attack. In *CCS*. ACM, 2007.

[109] Prateek Mittal and Nikita Borisov. ShadowWalker: Peer-to-Peer Anonymous Communication using Redundant Structured Topologies. In *CCS*. ACM, 2009.

[110] Prateek Mittal, Matthew Caesar, and Nikita Borisov. X-Vine: Secure and Pseudonymous Routing in DHTs Using Social Networks. In *NDSS*, 2012.

[111] Prateek Mittal, Matthew Wright, and Nikita Borisov. Pisces: Anonymous Communication using Social Networks. *arXiv preprint arXiv:1208.6326*, 2012.

[112] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *IMC*, 2010.

[113] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *CCS*, 2012.

[114] Michael Molloy and Bruce A. Reed. A critical point for random graphs with a given degree sequence. *Random structures and algorithms*, 6(2/3), 1995.

[115] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing Social Networks. In *Security and Privacy*, 2009.

[116] Giang Nyugen, Stefanie Roos, Thorsten Strufe, and Mathias Fischer. RBCS: A Resilient Backbone Construction Scheme for Hybrid Peer-to-Peer Streaming. In *LCN*, 2015.

[117] Andriy Panchenko, Stefan Richter, and Arne Rache. NISAN: Network Information Service for Anonymization Networks. In *CCS*, 2009.

[118] Christos H Papadimitriou and David Ratajczak. On a Conjecture Related to Geometric Routing. In *Algorithmic Aspects of Wireless Sensor Networks*. 2004.

[119] Thomas Paul, Stephen Stephen, Hani Salah, and Thorsten Strufe. The Student's Portal Ilmenau: A Holistic OSN's User Behaviour Model. In *PICCIT*, 2015.

[120] Radia Perlman. An Algorithm for Distributed Computation of a Spanningtree in an Extended LAN. *ACM SIGCOMM Computer Communication Review*, 15(4), 1985.

[121] Andreas Pfitzmann and Marit Hansen. A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. `http://www.maroki.de/pub/dphistory/Anon_Terminology_v0.34.pdf`, 2010.

[122] Bogdan Popescu. Safe and Private Data Sharing with Turtle: Friends Team-up and Beat the System. In *Security Protocols*, 2006.

[123] Swagatika Prusty, Brian Neil Levine, and Marc Liberatore. Forensic investigation of the oneswarm anonymous filesharing system. In *CCS*, 2011.

[124] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *ACM SIGCOMM*, 2001.

[125] Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 1998.

[126] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for Web Transactions. *TISSEC*, 1(1), 1998.

[127] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. *Computer Science*, 2003.

[128] Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *IEEE P2P*, 2001.

[129] Stefanie Roos. Analysis of Routing on Sparse Small-World Topologies. Master's thesis, Technische Universität Darmstadt, 2011.

[130] Stefanie Roos, Martin Beck, and Thorsten Strufe. Anonymous Addresses for Efficient and Resilient Routing in F2F Overlays. In *INFOCOM*, 2016 (to appear).

[131] Stefanie Roos, Giang T Nguyen, and Thorsten Strufe. Integrating Churn into the Formal Analysis of Routing Algorithms. In *NetSys*, 2015.

[132] Stefanie Roos, Florian Platzer, Jan-Michael Heller, and Thorsten Strufe. Inferring Obfuscated Values in Freenet. In *NetSys*, 2015.

[133] Stefanie Roos, Hani Salah, and Thorsten Strufe. Determining the Hop Count in Kademlia-type Systems. In *ICCCN*, 2015.

[134] Stefanie Roos, Benjamin Schiller, Stefan Hacker, and Thorsten Strufe. Measuring Freenet in the Wild: Censorship-resilience under Observation. In *PETS*, 2014.

[135] Stefanie Roos and Thorsten Strufe. Provable Polylog Routing for Darknets. In *HotPost*, 2012.

[136] Stefanie Roos and Thorsten Strufe. A Contribution to Analyzing and Enhancing Darknet Routing. In *INFOCOM*, 2013.

[137] Stefanie Roos and Thorsten Strufe. On the Impossibility of Efficient Self-Stabilization in Virtual Overlays with Churn. In *INFOCOM*, 2015.

[138] Stefanie Roos and Thorsten Strufe. Dealing with Dead Ends-Efficient Routing in Darknets. In *TOMPECS*, 2016 (to appear).

[139] Stefanie Roos, Liang Wang, Thorsten Strufe, and Jussi Kangasharju. Enhancing Compact Routing in CCN with Prefix Embedding and Topology-aware Hashing. In *MobiArch*, 2014.

[140] Hani Salah, Stefanie Roos, and Thorsten Strufe. Characterizing Graph-theoretic Properties of a Large-scale DHT: Measurements vs. Simulations. In *ISCC*, 2014.

[141] Hani Salah, Stefanie Roos, and Thorsten Strufe. Diversity Entails Improvement: A New Neighbour Selection Scheme for Kademlia-type systems. In *IEEE P2P*, 2014.

[142] Oskar Sandberg. Distributed Routing in Small-World Networks. In *ALENEX*, pages 144–155. SIAM, 2006.

[143] Benjamin Schiller, Stefanie Roos, Andreas Hoefer, and Thorsten Strufe. Attack Resistant Network Embeddings for Darknets. In *SRDSW*, 2011.

[144] Benjamin Schiller and Thorsten Strufe. GTNA 2.0-A Framework for Rapid Prototyping and Evaluation of Routing Algorithms. In *Summer Computer Simulation Conference*, 2013.

[145] Max Schuchard, Alexander W Dean, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. Balancing the Shadows. In *WPES*, 2010.

[146] Ayman Shaker and Douglas S Reeves. Self-stabilizing Structured Ring Topology P2P Systems. In *IEEE P2P*, 2005.

[147] Claude E Shannon. Communication Theory of Secrecy Systems. *Bell system technical journal*, 28(4), 1949.

[148] Dawinder S Sidhu. The Chilling Effect of Government Surveillance Programs on the Use of the Internet by Muslim-Americans. *University of Maryland Law Journal of Race, Religion, Gender and Class*, 7, 2007.

[149] Alistair Sinclair. Improved Bounds for Mixing Rates of Markov chains and Multicommodity Flow. *Combinatorics, probability and Computing*, 1(04), 1992.

[150] Atul Singh. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *INFOCOM*, 2006.

[151] Michael Sirivianos, Dirk Westhoff, Frederik Armknecht, and Joao Girao. Non-manipulable Aggregator Node Election Protocols for Wireless Sensor Networks. In *WiOpt*, 2007.

[152] Moritz Steiner, Taoufik En-Najjary, and Ernst W Biersack. Exploiting KAD: Possible Uses and Misuses. *ACM SIGCOMM Computer Communication Review*, 37(5), 2007.

[153] James PG Sterbenz, David Hutchison, Egemen K Çetinkaya, Abdul Jabbar, Justin P Rohrer, Marcus Schöller, and Paul Smith. Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines. *Computer Networks*, 54(8), 2010.

[154] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *ACM SIGCOMM Computer Communication Review*, 31(4), 2001.

[155] James V Stone. *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. JV Stone, 2013.

[156] Daniel Stutzbach and Reza Rejaie. Improving Lookup Performance Over a Widely-Deployed DHT. In *INFOCOM*, 2006.

[157] Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-Peer Networks. In *IMC*, 2006.

[158] Guanyu Tian, Zhenhai Duan, Todd Baumeister, and Yingfei Dong. A Traceback Attack on Freenet. In *INFOCOM*, 2013.

[159] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring Anonymity Revisited. In *Nordic Workshop on Secure IT Systems*, 2004.

[160] Eugene Vasserman, Rob Jansen, James Tyra, Nicholas Hopper, and Yongdae Kim. Membership-concealing Overlay Networks. In *CCS*, 2009.

[161] Liang Wang and Jussi Kangasharju. Measuring Large-scale Distributed Systems: Case of BitTorrent Mainline DHT. In *IEEE P2P*, 2013.

[162] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. Attacking the KAD Network. In *SecureComm*, 2008.

[163] Qiyan Wang, Prateek Mittal, and Nikita Borisov. In Search of an Anonymous and Secure Lookup: Attacks on Structured Peer-to-Peer Anonymous Communication Systems. In *CCS*, 2010.

[164] Di Wu, Ye Tian, and Kam-Wing Ng. Analytical Study on Improving DHT Lookup Performance under Churn. In *IEEE P2P*, 2006.

[165] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A Near-optimal Social Network Defense against Sybil Attacks. In *Security and Privacy*, 2008.

[166] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: Defending against Sybil Attacks via Social Networks. *ACM SIGCOMM Computer Communication Review*, 36(4), 2006.

[167] Guoxing Zhan, Weisong Shi, and Julia Deng. Design and Implementation of TARF: a Trust-aware Routing Framework for WSNs. *TDSC*, 9(2), 2012.

Appendices

Appendix A

Freenet Opennet Neighbor Selection

In this chapter, we describe our work with regard to the neighbor selection in the Opennet mode of Freenet. More precisely, we show that the current neighbor selection algorithm does not result in the desired topology through a measurement study in the real system. While the free selection of neighbors does not agree with our goals of a pure F2F overlay, our work helped to improve the efficiency of a highly used systems. So, despite the predominantly theoretical focus of this thesis, we also achieved some immediate practical improvements.

In the following, we describe the methodology for evaluating the impact of the neighbor selection algorithm in Freenet, then present the results of the measurements, followed by a discussion of potential improvements. The results have been published as part of [134] and have been integrated into the current version of Freenet [1].

A.1 Methodology

The goal was to find out if the distances between neighbors in the overlay actually follow the distribution from Kleinberg's model [86]. In addition, we measured the degree distribution, which influences the routing success observed in the system.

Upon establishing a connection, nodes exchange their own locations and the locations of their neighbors. Whenever the neighborhood changes, all neighbors are informed of the change. Hence, by logging all such messages, we obtained the degree of all neighbors of monitoring nodes and the distances between them and their neighbors.

Denote the measurement duration by T. We took snapshots of the neighborhood of our monitoring nodes each t time units. Let $G_k = (V_k, E_k)$ be a snapshot after $t \cdot k$ minutes for $k = 0 \dots K$ with $K = \lfloor T/t \rfloor$. The node set V_k consisted of our monitoring nodes M, the neighbors of nodes in M, and their neighbors. The subgraph G_k was induced, i.e., the edge set E_k consisted of all edges between nodes in V_k. We determined the empirical distance distribution of neighbors as the weighted average over all snapshots. Let $l(e)$ be the distance between the endpoints of edge e, and let the indicator function $\mathbf{1}_A(x)$ be 1 if $x \in A$ and 0 otherwise. Then the empirical distance distribution $\hat{L}$ was computed by

$$P(\hat{L} \leq x) = \sum_{k=0}^{K} \sum_{e \in E_k} \frac{\mathbf{1}_{[-\infty,x)}(l(e))}{\sum_{k=0}^{K} |E_k|}. \tag{A.1}$$

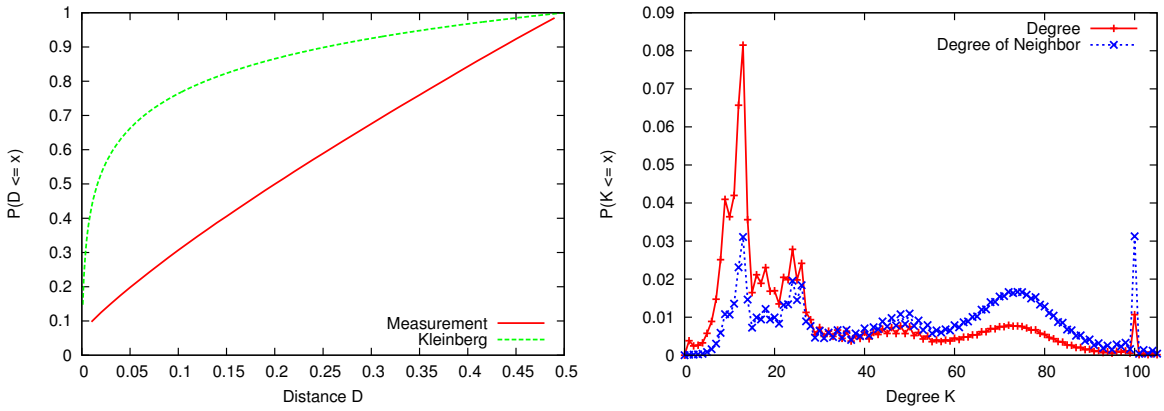
When obtaining the degree distribution, our own nodes might not represent a good sample for the average user with regard to bandwidth and uptime. Since both influence the degree of a node, we only considered the sets $N_k(m) \setminus M$ of neighbors of $m \in M$ at time $t \cdot k$. Let $deg(v)$ denote the degree of a node v. Analogously to the distance distribution, the empirical degree distribution of neighbors $\hat{D}'$ was then obtained as [2]

$$P(\hat{D}' = x) = \sum_{k=0}^{K} \sum_{m \in M} \sum_{v \in N_k(m) \setminus M} \frac{\mathbf{1}_x(deg(v))}{\sum_{k=0}^{K} \sum_{m \in M} |N_k(m) \setminus M|}. \tag{A.2}$$

Then, note the probability of being a neighbor of a node is proportional to the degree of a node. If the degree distribution of the network is D, the degree distribution D' of randomly chosen neighbors is given

[1] `https://github.com/freenet/fred/commit/3172c2865e0cbc0b198bdf1354a1340600f054e9`

[2] It is intended that nodes in the intersection of two neighborhoods are counted multiple times



(a) Distance (b) Degree

Figure A.1: Distance distribution of neighbors, degree distribution, and the degree distribution of neighbors

by

$$P(D' = x) = \frac{xP(D = x)}{\mathbb{E}(D)}. \tag{A.3}$$

Our measurements provided the empirical degree distribution of neighbors $\hat{D}'$. So an empirical degree distribution $\hat{D}$ was obtained by solving a system of linear equations based on Equation A.3. Let d_m denote the maximal observed degree. The system of linear equations consisted of $d_m + 1$ equations with $d_m + 1$ variables $P(\hat{D} = x)$ for $x = 1 \dots d_m$ and $\mathbb{E}(\hat{D})$. The first d_m equations were derived from transforming Equation A.3 to $xP(D = x) - P(D' = x)\mathbb{E}(D) = 0$. The last equation used that $\hat{D}$ is a probability distribution, so that $\sum_{x=1}^{d_m} P(\hat{D} = x) = 1$. The system of equations thus could be solved using Gaussian elimination.

A.2 Set-up and Results

The data for this analysis was obtained from a two week measurement in May 2013 using 12 instrumented Freenet clients.

Figure A.1a shows the cumulative distance distribution observed in our measurements in comparison to the function $1/d$ for $d > 0.01$. Indeed, each node had a high number of close neighbors. However, contacts at distance exceeding 0.05 seemed to be chosen uniformly at random, as indicated by the linear increase of the distribution function.

With regard to the degree distribution, there are several peaks in the degree distribution around 13, 50, 75, and 100 (cf. Figure A.1b). Indeed, these seem to correspond to typical bandwidths, e.g., for 2 Mbit/s 100 neighbors are allowed. Note that we observed nodes with a degree of up to 800, but nodes with a degree of more than 100 make up less than 1 % [3]. Nodes with a degree of less than 10 are likely to be in the start-up phase since by default a node is allowed at least 14 neighbors.

A.3 Discussion

We have seen that nodes have a high number of close neighbors. These are probably found by announcements sent via the seed nodes and routed towards a node's own location. However, the long-range contacts are chosen uniformly at random, i.e., with a probability proportional to $\frac{1}{d^0}$, i.e., independently of their distance, rather than with probability of $\frac{1}{d^1}$. The routing complexity when nodes are connected independently of their distance is of order $n^{2/3}$ [86].

To illustrate the impact of such an non-optimal neighbor selection, we performed a simulation study of the Freenet routing algorithm.

We generated a ring topology with 15,000 nodes. Each node was assigned a random location in $[0, 1)$, corresponding to Freenet's key space. Each node was connected to the k closest nodes on the ring. In

[3] since May 21st, 2013, a week after our measurement, the maximal degree is actually set to be 100, see `https://github.com/freenet/fred/commit/c85319999cfe85369c6f4e92fb14efd769c60a59`

addition, for each node a random integer l was chosen according to the empirical degree distribution we observed in the Freenet network. The node was then given $d = max\{l-2k, 0\}$ long-range contacts chosen proportional to $1/d^r$ for $r = 0$ (independent of the distance as in Freenet) and $r = 1$ (anti-proportional to the distance suggested by Kleinberg).

The average routing length was less than 13 hops for an optimal distance distribution ($r = 1$), but 37.17 hops for $r = 0$, i.e., the distance distribution we found in Freenet. When connecting each node to the 3 closest nodes on the ring, i.e., $k = 3$, the average routing length for $r = 0$ decreased to 28 because progress was made using the additional short-range links, but the average routing length for $r = 1$ increased by 30% to 17 hops. These results show that Freenet's performance can be drastically improved by, e.g., dropping and adding connections based on the distance of node identifiers.

A Kademlia-like bucket system [104] could be used to achieve the desired distance distribution while still allowing a wide choice of neighbors. So, the decision of dropping a neighbor can be made both on its performance and its location. The number of buckets of the number of contacts per bucket and hence the degree can be chosen dependent on the bandwidth a node contributes to the system, in order to retain this incentive of the current neighbor selection scheme. An alternative approach can be to include Opennet in the location swapping algorithm used by Darknet nodes, which has been shown to achieve a Kleinberg-like distance distribution in [142] for a static network. An in-depth simulation study is required to give concrete guidelines.

Appendix B

Inferring Obfuscated Values in Freenet

We evaluate the anonymization of node statistics in Freenet. As such our required functionalities are realized independently of statistics, we did not include the consideration of Freenet's statistics algorithm in the main part of the thesis. However, a real-world prototype requires such statistics to monitor the system for detecting unanticipated scenarios and attacks. As a consequence, we evaluated of the current algorithms for obtaining such statistics provide sufficient anonymity, i.e., prevent that the sensitive information of individual nodes in the system can be derived. We found that indeed an active local adversary can infer sensitive values about its neighbors and suggested improvements. The results have been published in [132].

In the following, we first introduce some background on i) Freenet's statistic collection, and ii) Bayesian learning, as we use the latter to infer the sensitive information. Afterwards, we present our attack on the anonymity of the statistics and evaluate it both through simulation and real-world measurements. We conclude this chapter by suggesting an improved anonymization algorithm.

B.1 Background

We start by explaining how and which node statistics are collected in the Opennet mode of Freenet. Afterwards, we give a short introduction to Bayesian learning.

B.1.1 Freenet's Anonymous Statistics

In addition to content discovery and maintenance, Freenet offers the possibility to obtain global statistics about the network, such as the uptime, bandwidth, and link length distribution. These statistics are mainly intended to be used by the developers for monitoring the system and improving the algorithms based on the results. Monitoring the system has been proven necessary to counteract large-scale attacks and undesired side effects from code changes[1]. The statistics are supposed to be sampled uniformly at random from all nodes in the network in an anonymous fashion, hiding both the node providing a value and the provided value.

These requirements are realized by sending messages of the type `ProbeRequest` into the network, inquiring one certain property, e.g., the uptime, of a randomly selected node. The uniformity of the sample is provided by a random walk with Metropolis-Hastings correction [45]. More precisely, a node u, which receives a `ProbeRequest`, first probabilitistically decreases the htl. If the htl is 0, it replies with an obfuscated value, otherwise it selects a random neighbor v with $deg(v)$ neighbors. The message is then forwarded to v with probability $f = \min\{1, \frac{deg(u)}{deg(v)}\}$. With probability $1 - f$, the message remains at u and u repeats the process. In this fashion, the bias towards high-degree nodes of random walks is counteracted, allowing a close to uniform selection. The identity of the responder is obfuscated by the probabilistically decreasing htl. By default, the initial htl of a `ProbeRequest` is set to 70, which is then decreased by 1 in each step of the random walk. If a message with $htl = 1$ is received, the htl is only decreased with a probability $p = 0.2$. In order to prevent timing analysis, the response is delayed, such that it is unclear from which node the response origins [2].

The value of the response is obfuscated to prevent reidentification. The actual value x of the responding node u is multiplied by a factor m, which is chosen according to a normally distributed random variable with mean 1 and variance σ, cutoff at 0.5 and 1.5. In other words, u generates a standard normal

[1] `http://draketo.de/english/freenet/meltdown-2014-04`
[2] `https://wiki.freenetproject.org/FCPv2/ProbeRequest`

distributed random variable r and computes its answer $\tilde{x} = x \cdot m$ with

$$m = \begin{cases} 0.5, & r\sigma + 1 < 0.5 \\ r\sigma + 1, & 0.5 \leq r\sigma + 1 \leq 1.5 \\ 1.5, & r\sigma + 1 > 1.5 \end{cases}. \tag{B.1}$$

However, if a large number of requests are sent to a neighbor with $htl = 1$, the answers might still reveal information about the actual value b of that neighbor. The number of `ProbeRequests` a node is allowed to receive from the same neighbor is bound by 10 per minutes, but as we show in further sections, good predictions are already possible when having only a few hundred of responses. We make use of a standard statistical learning algorithms from the field of Bayesian Statistics.

B.1.2 Bayesian Statistics

The main idea of Bayesian Statistics is to infer a hidden value x based on observations $o_1, \ldots, o_n$, which are dependent to x. In our case, the hidden value x is the property of a Freenet node, and the observations are answers to `ProbeRequests`. In general, Bayesian Statistics is an application of Bayes' Rule, which we state for continuous random variables in the following, before shortly explaining its role in Bayesian Statistics. For a detailed introduction on Bayesian Statistics including proofs of Equations B.2 and B.3, see e.g.,[155, 92].

Let X be the random variable for the hidden value and O_i be the i-th observation. Observations are identically distributed according to a random variable O and independent of each other given X. To simplify the notation, we assume X and O to be continuous with density functions f_X and f_O. Let $f_{X,O}$ denote the common density of X and O, and $f_{X|O=o}$ be the density of X given the observation o with $f_O(o) > 0$. Then Bayes' Rule states that

$$f_{X|O=o}(x) = \frac{f_{O|X=x}(o)}{\int f_{O|X=y}(o) f_X(y) dy} f_X(x). \tag{B.2}$$

and because the observations are independent given X,

$$\begin{aligned} & f_{X|O_n=o_n,\ldots,O_1=o_1}(x) \\ =& \frac{f_{O|X=x}(o_n) f_{X|O_{n-1}=o_{n-1}\ldots,O_1=o_1}(x)}{\int f_{O|X=y}(o_n) f_{X|O_{n-1}=o_{n-1},\ldots,O_1=o_1}(y) dy}. \end{aligned} \tag{B.3}$$

The function $f_{O|X=x}(o)$ is called the likelihood function and is denoted by $L_x(o)$ for continuous random variables, while it is $L_x(o) = P(O = o|X = x)$ for discrete O.

In Bayesian Statistics, we aim to approximate the distribution X of the hidden value. The mostly likely value according to X is then chosen as a guess for the hidden value x. Starting with an initial guess f_X^0 of X's density, the i-th guess f_X^i for $i = 1 \ldots n$ is obtained by

$$f_X^i(x) = g_x(o_i) f_X^{i-1}(x).$$

Note that $f_X^i(x)$ can be either the density or the probability of x, depending if X is continuous or discrete. In our analysis, we choose g_x to be equal to the normalized likelihood function, i.e.

$$g_x(o) = \frac{L_x(o)}{\int L_y(o) f_X^{i-1}(y) dy}.$$

Hence, $f_X^i(x)$ is computed as

$$f_X^i(x) = \frac{L_x(o)}{\int L_y(o) f_X^{i-1}(y) dy} f_X^{i-1}(x). \tag{B.4}$$

More complex functions g_x, usually weighting observations, can be applied to avoid overfitting, but did not improve the results in our evaluation. The design of our inference algorithm, presented in the next section, mainly consists of determining the likelihood function.

B.2 Inference Algorithm

In this section, we show how a hidden value of a neighboring node can be inferred. After deriving a stochastic model of the obfuscation process, we first obtain f_X^0 as an approximation of the global distribution G of the queried property. Then we determine the likelihood function required by Equation B.4 to determine $f_X^1, \ldots f_X^n$ and infer the hidden value.

B.2.1 Algorithm Overview

Before deriving the individual components of our inference algorithm, we here list the steps of the algorithm:

1. Send t `ProbeRequest`s with the maximal htl in the network and gather answers
2. Compute an approximation of the global distribution G of the desired property from the m answers (as detailed in Section B.2.3)
3. Select a target node whose hidden value should be inferred and set $f_X^0 = G$
4. Send n `ProbeRequest`s with $htl = 1$ to the target node
5. Compute $f_X^1, \ldots, f_X^n$ based on Equation B.4 with the likelihood function derived in Section B.2.4
6. Infer the hidden value of the targeted node as the value with the highest likelihood according to f_X^n

B.2.2 Modeling Obfuscation

In this section, we give a general model of the obfuscation process when querying a neighbor using a `ProbeRequest` with $htl = 1$. Let B be the set of possible hidden values and A the set of possible answers. Note that B could be any subset of $\mathbb{R}$, countable or uncountable, while A is always uncountable, usually an interval. With probability p, we receive an obfuscation $b \cdot m$ of the hidden value $b \in B$ with m chosen as by Equation B.1 using noise σ. With probability $1-p$, a random node answers with an obfuscated value. In the following, denote the density and the distribution function of the standard normal distribution by ϕ and Φ, respectively. For any value $a \in A$, let $c(a)$ be a boolean denoting the fact that there exists a value $b \in B$ with potentially $P(G = b) > 0$, i.e., G is discrete, and $a \in \{0.5b, 1.5b\}$, i.e.,

$$c(a) = \begin{cases} true, & G \text{ discrete}, \exists b \in B, a \in \{0.5b, 1.5b\} \\ false, & \text{otherwise} \end{cases}.$$

If $c(a) = true$, the value a can be attained with strictly positive probability due to the cutoff in Equation B.1 for some b. More precisely, if b is the hidden value, the probability for a to be an answer is $\Phi(-0.5/\sigma)$. If, on the other hand, $c(a) = false$, there is no value b for which a is obtained with positive probability. Thus, the likelihood function treats the two types of observations differently.

In the following, we need the likelihood $obf_b(a)$ that a value $b \in B$ is obfuscated to $a \in A$.

Theorem B.1. *The likelihood function $obf_b(a)$ of a being an obfuscation of b is given by*

$$obf_b(a) = \begin{cases} \Phi\left(\frac{-0.5}{\sigma}\right), & a \in \{0.5b, 1.5b\} \\ \phi\left(\frac{a/b-1}{\sigma}\right), & a \in (0.5b, 1.5b), c(a) = false. \\ 0, & \text{otherwise} \end{cases}$$

Proof. Consider that the standard normal distributed random number r, which relates the hidden value b and the observed answer a, is equal to $\frac{a/b-1}{\sigma}$ if $a/b \in (0.5, 1.5)$ by Equation B.1. If a/b is below 0.5 or above 1.5, a cannot be an answer generated from b, and if a/b is either -0.5 or 0.5, r could have been any number less or equal to $-0.5/\sigma$ or greater or equal to $0.5/\sigma$.

For the proof, first assume that $c(a) = true$. Then there exists $b' \in B$ with $P(G = b') > 0$ such that $a = 0.5b'$ or $a = 1.5b'$. If $b' = b$, then the probability to generate a from b is equal to $\Phi\left(\frac{-0.5}{\sigma}\right)$. If $b' = b$, a can be an answer with a strictly positive probability for some value b' but is almost surely not chosen as an answer for b. Hence the probability that a is an answer generated by a hidden value b is 0. Now, let $c(a) = false$. Then it is not an answer generated with a non-zero probability and only density functions have to be considered. If a/b is between 0.5 and 1.5, the density function of a given b is the density of a normal distribution with mean 1 and variance σ, i.e. $\phi\left(\frac{a/b-1}{\sigma}\right)$. □

In order to simplify the notation, we use the integral to denote integration and summation in the following. In particular, if a random variable H is continuous with density γ, we have that for any real-valued function f,

$$\int_B f(x)\, dP_H(x) = \int_{-\infty}^{\infty} f(x)\gamma(x)dx.$$

In contrast, for a discrete random variable H with values in B, we get

$$\int_B f(x)\, dP_H(x) = \sum_{x \in B} f(x)P(H = x).$$

B.2.3 Global Distribution

An approximation of the global distribution is obtained by collecting responses $o_1, \ldots, o_t$ to t queries sent to random nodes. For each response o_i, we consider the likelihood that it was generated by b for all $b \in B$. Formally, let $f_G(b)$ denote the approximated density or probability function of G. We compute

$$f_G(b) = \frac{1}{t} \sum_{i=1}^{t} \frac{obf_b(o_i)}{N_i} \tag{B.5}$$

where $N_i = \int_B obf_x(o_i)dx$ is a normalization constant.

B.2.4 Likelihood Function

Given n answers to requests, the distribution of the hidden value is step-wise adjusted using Equation B.4. The likelihood function is the decisive component of the inference algorithm.

Theorem B.2. *For all $a \in A$ and $b \in B$,*

$$L_b(a) = p \cdot obf_b(a) + (1-p) \cdot \int_B obf_x(a) dP_G(x). \tag{B.6}$$

gives the likelihood to receive an answer a if the target node has hidden value b.

Proof. Let E denote the event that the targeted node answers and recall that $E^\perp$ denotes the complement of E. Furthermore, let O and X be the reply and the hidden value, respectively. Note that for discrete random variables O and X,

$$\begin{aligned} L_b(a) = P(O = a | X = b) &= \frac{P(O = a, X = b)}{P(X = b)} \\ &= P(O = a | X = b, E) P(E) + P(O = a | X = b, E^\perp) \left(1 - P(E)\right). \end{aligned}$$

The last step holds because

$$\frac{P(O = a, X = b, E)}{P(X = b)} = \frac{P(O = a | X = b, E) P(X = b, E)}{P(X = b)}$$

and the targeted node answers with probability $P(E) = p$, independent of the value of X. Analogously, for continuous O

$$L_b(a) = f_{b,E}(a) P(E) + f_{b,E^\perp}(a) \left(1 - P(E)\right)$$

holds. Hence, the likelihood function can be expressed as

$$L_b(a) = p \cdot own_b(a) + (1-p) \cdot other_b(a). \tag{B.7}$$

It remains to derive $own_b(a)$ and $other_b(a)$. $own_b(a) = obf_b(a)$ follows from the definition of $obf_b(a)$ being the likelihood that a value b is obfuscated to a. The term $other_b(a)$ gives the probability that a random node answers, and is hence independent of the hidden value of the targeted node. Rather, the probability of generating a value a is computed by summation of all possible actual values x of an answering node and the probability to generate the answer a from x, i.e. $other_b(a) = \int_{b \in B} own_x(a) dP_G(x)$. This completes the proof. □

B.3 Evaluation

We evaluated the inference algorithm both by simulating `ProbeRequests` in a synthetic network and by real-world measurements. Both evaluations share the same scenario, use the same metrics, and are based on the global bandwidth distribution observed in Freenet, which we describe before the actual evaluation.

B.3.1 Scenario

For the evaluation, we focus on the bandwidth. Inferring the bandwidth can help to detect bridge nodes, which have neighbors participating in the Darknet as well as neighbors participating in the Opennet. They are hence particularly important for connecting Opennet and Darknet, and are likely targets of

attacks. Discovery of such bridge nodes can be achieved by making use of the fact that the maximal number of neighbors of a Freenet node with upload bandwidth b (in KiB/s) is bound by

$$maxdeg(b) = \max\{10, \min\{100, \sqrt{b * 12}\}\}. \tag{B.8}$$

If a node accepts less Opennet neighbors than the maximum number over a longer time, it is likely to have connections into the Darknet. Hence obfuscating the bandwidth is an essential protection against attacks on important nodes. Note that the bandwidth dedicated to Freenet is usually much less than the total bandwidth of the node, so that detecting the total bandwidth is not useful. In the current Freenet implementation, the response probability is $p = 0.2$, and the noise is $\sigma = 0.05$ for the bandwidth. The set $B \subset \mathbb{N}$ is the set of all possible bandwidths. The set A of potential obfuscated values is $A = [0.5 \min B, 1.5 \max B]$ by Equation B.1.

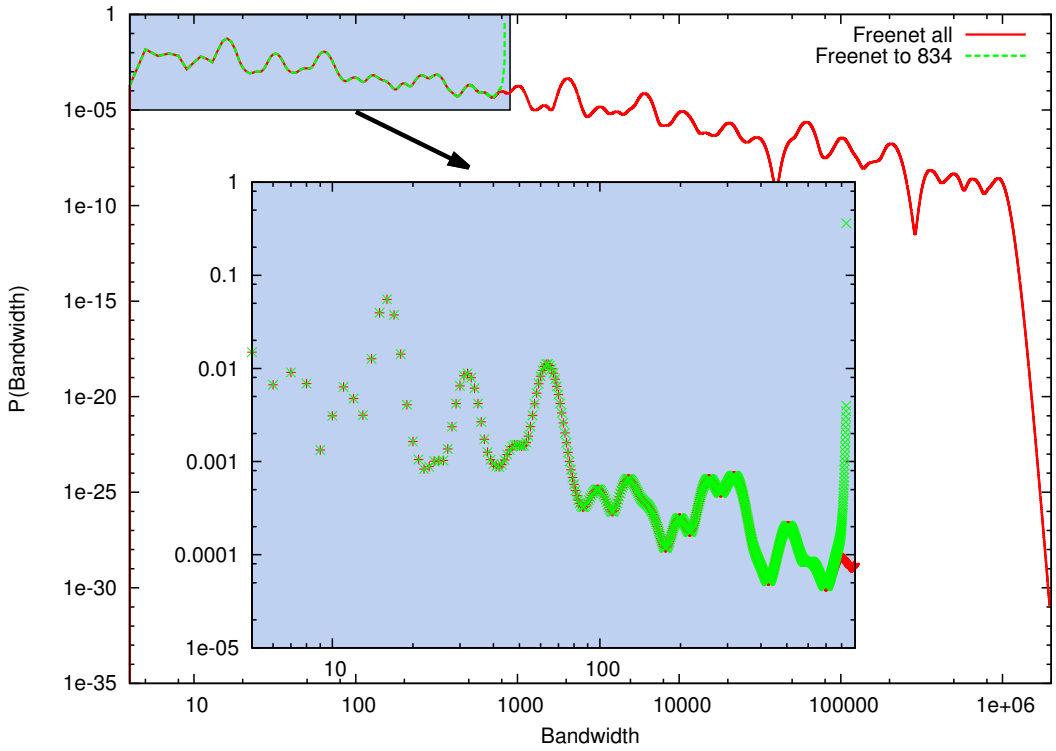


Figure B.1: Inferred global bandwidth distribution in Freenet

B.3.2 Metrics

In the following, denote the inferred value by $\tilde{x}$, i.e.

$$\tilde{x} = argmax_{x \in B} f_X^n(\tilde{x})$$

is the value with the maximal probability according to the distribution of the hidden value X considering n answers.

Our evaluation considered the error between the inferred value $\tilde{x}$ and the actual value x, indicating how close our algorithm is to the correct value. Both the absolute error $|\tilde{x} - x|$ and the relative symmetric error $\frac{2|\tilde{x}-x|}{x+\tilde{x}}$ were considered. The latter is of particular importance as the higher x is, the higher an absolute error can be tolerated. Thus, the success rate within factor r, i.e., the fraction of inferred values $\tilde{x}$ within $[x - rx, x + rx]$, was evaluated as well. Considering sufficiently close values as successful inferences is well-reasoned because an accurate prediction was not required to judge a node's capabilities such as its available bandwidth. In general, the inference of accurate values was important for low bandwidths, but both hard to achieve (due to the increased range for the obfuscated values) and of limited use for bandwidths in the order of hundreds of thousands of KiB/s. In the light of the detection of Darknet neighbors, we in particular considered the success and error with regard to the maximal number of neighbors $maxdeg(\tilde{x})$ and $maxdeg(x)$.

In addition, we analyzed the certainty in the prediction as well as in the actual value. The certainty might help to identify incorrect predictions, which possibly show a low certainty. More precisely, we

defined the certainty of any $b \in B$ to be $cer(b) = f_X^n(b)$, and the certainty within factor r to be $cer_r(b) = \sum_{b' \in B \cap [b-rb, b+rb]} f_X^n(b')$. We distinguished between the certainty within factor r with regard to true-positives (TP), i.e., $cer_r(x)$ for $\tilde{x} \in [x - rx, x + rx]$, the certainty with regard to false-positives (FP), i.e., $cer_r(\tilde{x})$ for $\tilde{x} \notin [x - rx, x + rx]$, and the certainty with regard to false-negatives (FN), i.e., $cer_r(x)$ for $\tilde{x} \notin [x - rx, x + rx]$. The factor r is introduced to account for the fact that for high values of x, it is harder to distinguish between x and a value close to x, hence reducing the certainty in one concrete value but still achieving a high certainty on the set of close values for a reliable prediction.

B.3.3 Data Sets

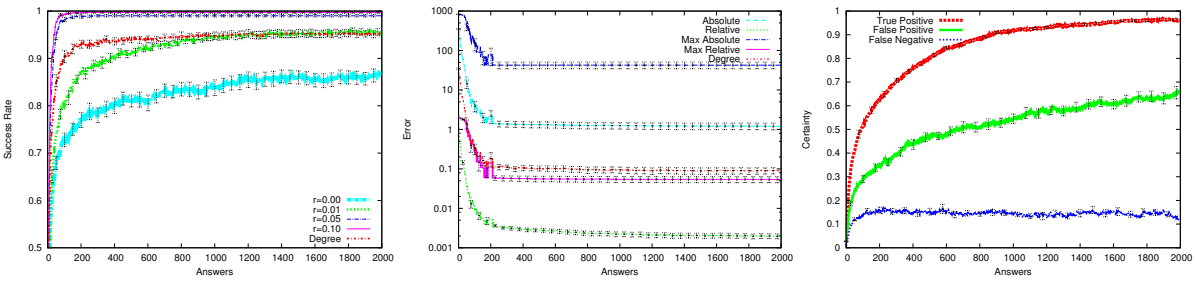
The same global distribution was used for both simulations and measurements, in order for the simulations to be realistic. We collected roughly 13,000 answers to `ProbeRequest`s with $htl = 70$ in the real network. Because the maximal number of neighbors is bound by 100 in Equation B.8, only bandwidths up 834 KiB/s, the lowest bandwidth for having 100 neighbors, were considered. All replies above $1.5 \cdot 834$ were treated as $1.5 \cdot 834$ to reduce the computation complexity. We obtained the global distribution G as described in Section B.2.3. The result is displayed in Figure B.1, both for the complete distribution and for the cutoff. Values of more than 1 GiB/s (roughly 10^6 KiB/s) were observed. Due to the option of configuring the bandwidth used for Freenet manually, these values might result from (accidental or intended) misconfiguration.

B.3.4 Simulation

The inference algorithm was first evaluated in a simulation study. Due to the bound number of requests nodes answer per minute, measurements were time- and resource-consuming. Hence, a simulation study allowed for more extensive testing and fine-tuning of the parameters, most importantly the number of answers n needed for a reliable inference.

Simulation Design The simulation consisted of two steps: i) generation of network topology and executing of `ProbeRequest`s within the synthetic topology, and ii) application of the inference algorithm described in Section B.2 to the simulated answers.

For i), we constructed a network according to the parameters k, the network size, and the global bandwidth distribution G. We first created a ring of k nodes, and assigned each node u a bandwidth $bw(u)$ according to G. Based on Equation B.8, we determined the maximal number of additional links $maxdeg(u) - 2$ a node could have. These additional links were created by adding each node u to a list $maxdeg(u) - 2$ times. Then we removed two random entries u and v from the list and added an edge between them if $u \neq v$ and u and v had not been connected before. The algorithm terminated when less than two elements remained. According to [114], the degree distribution of the obtained random graph should be roughly close to the desired degree distribution. Note that the underlying ring structure guaranteed that the network was connected. According to recent measurements, Freenet's topology is indeed close to a ring with additional random links, as shown in Appendix A.



(a) Success Rate (b) Error (c) Certainty $r = 0.0$

Figure B.2: Success ratio of inferring hidden value, error rate between inferred value and actual value, success rate, and certainty in the inference with regard to the number of considered answers.
To improve readability, variances are plotted every 50 steps for the success rate and the error, starting at either 25 (maximal error, r=0.01 and r=0.05) or 50, and at each 75 steps for the certainty, starting at 25 (TP), 50 (FP), and 75 (FN)

For executing the inference algorithm, the simulation implemented the `ProbeRequest` as described in Section B.1.1 with parameters p and σ. Samples of the global distributions were obtained by choosing

w monitoring nodes and executing a total of t requests with $htl = 70$, each originating from a randomly chosen monitoring node. The target node was randomly selected and n `ProbeRequest`s were sent with $htl = 1$. The inference algorithm was implemented as designed in Section B.2.

Set-up For the actual simulation, 20 runs were executed as follows:

1. Construct a network with parameters $k = 5000$ nodes and G as derived in Section B.3.3
2. Obtain an approximation of G based on $t = 10,000$ requests
3. Infer the bandwidth of 50 randomly selected target node based on $n = 2,000$ samples.

The parameters $p = 0.2$ and $\sigma = 0.05$ were applied for the obfuscation, as in the original Freenet code. The factor r for considering an inference to be successful was varied within $\{0.0, 0.01, 0.05, 0.1\}$.

Expectations Our inference algorithm should achieve a high success rate, but success could not be guaranteed due to its stochastic nature and its lack of considering the network topology. More precisely, incorrect predictions could happen if the fraction of neighbors of the targeted node with the same bandwidth b is high, resulting in a lot of replies pointing to b rather than the actual victim bandwidth. However, such scenarios were extremely unlikely, because all bandwidths had a much lower global probability than 0.2 as can be seen from Figure B.1. Incorrect inference due to the bandwidth distribution in the target node's neighbor should be rare.

The performance of the algorithm was expected to improve with the number of observed answers n. Thus, we expected that success ratio and certainty increased with the number of observations, while the error decreased. The increase in performance was expected to be most noticeable when the number of observations is low, and converge towards a steady state, optimally a perfect inference, when n increases. The certainty was bound to improve slower than the success rate and the error. After first guessing the correct value, more answers were needed until the inference was considered reliable.

The success ratio and certainty should improve with the factor r because the number of values which are considered to be correct increased. An accurate inference of high bandwidths was unlikely because the multiplication with a normally distributed factor produces a larger range of values than for a low bandwidth. However, the inferred value should be close in relation, such that even for a low r of 0.01 or 0.05, we expected a drastically higher success rate and certainty than for $r = 0.0$. The increase was likely to be more pronounced for the certainty than for the success rate. We expected values close to actual value to have a high certainty as well, especially for high bandwidths.

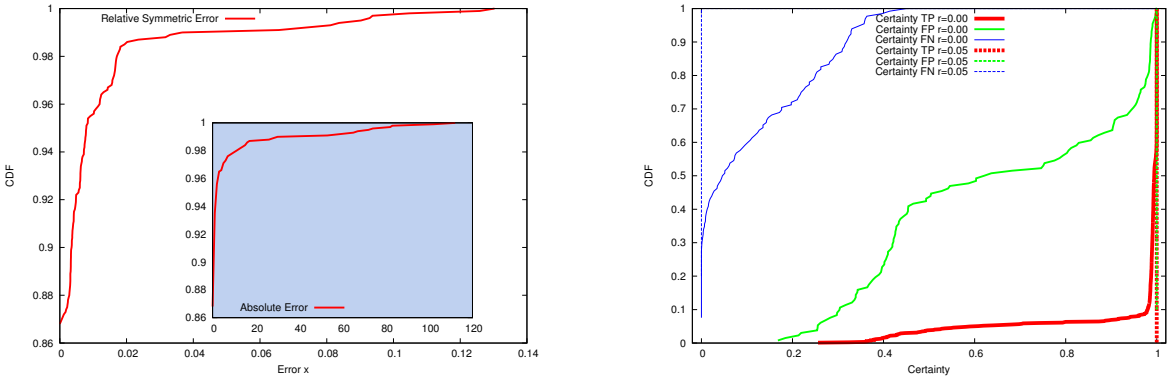
For a more detailed analysis, we were also interested in the distribution of error and certainty over all experiments. The relative error was expected to be mostly small stemming from a small inaccuracy of the inference for high values. The certainty for true positive inference was expected to be higher than for false positives, in the best case enabling the detection of false positives.

Results Figure B.2 displays the evolution of the success rate, error, and certainty considering 1 to 2000 answers. First of all, the success ratio of our algorithm was in general high. Secondly, the number of answers needed to obtain good predictions was mostly less than 200, as can be seen from the strong initial increase in the success ratio displayed in Figure B.2a. So, all in all our simulation indicated that our algorithm is both accurate and efficient.

We now discuss the results in more detail: For an accurate prediction, i.e., $r = 0.0$, the maximum success rate was roughly 86.3%. Already by increasing r to 0.01, the success rate was improved to 95.6%, showing that most incorrectly inferred values only are off by a very slight amount. The success rate for inferring the maximal degree according to Equation B.8 was similar to $r = 0.01$ when considering all 2000 queries. However, less answers were needed for a good inference when considering the degree, because the acceptable error scales like $b^{1/2}$ for the degree the correct value b , but linearly with b for $r = 0.01$. Slight errors for low bandwidths were thus accepted for the degree, but not for $r = 0.01$. When increasing r to 0.05 or even 0.1, the success rate converges to a constant value of 99% and 99.6 % within 150 answers. So, inferring the hidden value within close bounds is possible even with a rather low number of samples.

The results for the absolute and relative error as displayed in Figure B.2b were in agreement with the results for the success ratio. The error decreased rapidly initially, then reached a steady state at about 200 answers. The final average mean relative symmetric error was around 0.002, and the average maximum below 0.1. The cumulative distribution of the relative error in Figure B.3a considering all 1000 trials distributed over 20 runs shows that the overall maximum was approximately 0.13. An relative error of more than 0.02 was rare, less than 1.5% of the trials achieved such an error. The distribution of the absolute error was similar, only on a different scale, reaching value of up to 112, for an incorrect inference of 834 rather than 722.

The certainty in the inferred or correct value increased slower than the success rate (Figure B.2c), as expected. The certainty in correctly inferred values was higher (TP) than for incorrectly inferred values (FP) for $r = 0.0$. However, slight errors were frequent for high hidden values. The certainty then was mainly distributed between values close to the hidden value, such that the certainty in each individual value was low. Hence, a low certainty was usual for high values, regardless if the inference was accurate or not. For $r > 0$, the difference between true and false positives vanished, as shown in Figure B.3b for $r = 0.05$ in addition to $r = 0.0$. For $r = 0.05$, the cumulative certainty of all values that were considered correct was 1, regardless if the inference was correct or not. As a consequence, a low certainty could not be used to differentiate between correct and incorrect inference.



(a) Error (b) Certainty

Figure B.3: Distribution of the absolute and relative error as well as the certainty after 2000 answers

B.3.5 Measurements

In this section, we evaluated our algorithm within the real network, in which joining and leaving nodes, as well as rejected or failed probes complicated the inference.

Design We inserted two nodes into Freenet. We adapted the Freenet code in order to send `ProbeRequest` to a specific neighbor rather than randomly. The first node, the requester v, sent `ProbeRequest`s to the the target node u. Both nodes participated in the Opennet, hence they obtained a diverse and ever changing neighborhood, but maintained a Darknet connection between them. Note that the Darknet connection was not required for inferring the bandwidth of a specific node. Due to the structure of the Opennet topology, a node would have connected to a targeted node soon if it had chosen a node identifier close to the target's and dropped connections to other nodes, as shown in [158]. We merely reduced the time needed for the measurements by considering Darknet connections, but the results apply for an Opennet connection between v and u as well.

Nodes could reject `ProbeRequest`s, i.e., an error message was returned rather than a value, if they were overloaded or already received more than 10 requests from the same node within the last minute. We thus limited our requests to 10 per minute. The first request was sent 10 to 20 minutes after the nodes were started, allowing them to gain an reasonable sized set of at least 5 neighbors.

Set-up Freenet clients with version Freenet 0.7.5 Build # 1465 were used for the measurements. The nodes' bandwidth was varied between 16, 64, 85, and 2000 KiB/s. While 16, 64, and 2000 KiB/s were frequent bandwidths, 85 was used to test the performance of the algorithm for an uncommon bandwidth. Furthermore, we used the restricted global distribution G, such that 2000 KiB/s should be inferred as 834 or more. We sent 1000 `ProbeRequest`s per run, usually 5 to 9 each minute, of which at least 462 and on average 871.25 were answered. The remaining requests were rejected or dropped. Our simulations showed that the error stabilized after about 200 answers, so that the sampled data should be sufficient.

Expectations Given the results achieved in our simulation, we expected a high success rate, especially for a comparable low bandwidth of 16. Similarly, 2000 KiB/s should always be accurately to be predicted to be 834, i.e., any bandwidth allowing 100 neighbors. For 64 and 85 KiB/s, we expected the inference to be mostly accurate, possibly deviating from the actual value slightly for some runs, as observed in the simulations.

x	**Error**	Certainty TP	Certainty FP	Certainty FN
16	0:10	1.0 [1.0,1.0]	-	-
64	0:10	0.998 [0.992,1]	-	-
85	0:8,1:2	0.846 [0.518,1]	0.902 [0.901,0.903]	0.098 [0.097,0.099]
2000	0:10	1.0 [1.0,1.0]	-	-

Table B.1: Absolute error frequency and certainty in true positive (TP), false positive (FP), and false negative (NP) inferences in form of mean [min,max] for 10 measurements

Results Table B.1 lists all measured absolute errors and the certainty in the results. An accurate inference was achieved in 100% of the runs for 16 , 64, and 2000KiB/s, while for 85 KiB/s the success rate was only 80%. However, for the 2 inaccurate inferences, a value of 86 rather than 85 was inferred. The results were in agreement with the simulation results, allowing an accurate inference for lower bandwidths and slight inaccuracies for high bandwidths. The 100% accuracy for 2000KiB/s was to be expected since it was classified as being 834 or more. The certainty (using $r = 0.0$) in the inferred value was 1 or close to 1 for 16, 64, and 2000 KiB/s, i.e., the bandwidth with 100% success rate. The certainty in the inference for 85 KiB/s varied between slightly above 0.5 and 1 for the true positives, whereas the certainty for the two false positives was roughly equal to the median of the true positive inference, being just above 0.9. Though the result was not significant, it agreed with our observation in Section B.3.4 that there was no simple criterion for distinguishing true and false negatives. The certainty for 84, 85, and 86 combined was above 0.99 for all runs, so that the algorithm could indeed always closely identify the range of the actual value.

We have seen that the inference is effective both in simulations as in real-world measurements. In the following, we discuss a potential protection mechanism.

B.4 Discussion

We have seen that the obfuscation of the node statistic collection in Freenet can be broken by sending multiple probes to the same neighbor.

We suggest an improvement on the value obfuscation. Increasing the noise σ is bound to impair the accuracy of the inference in practice, but does not guarantee anonymity and also impairs the accuracy of the global statistics. We hence suggest the use of 'shadow values': Each node u keeps $k-1$ random values chosen according to a globally known distribution H. When answering a `ProbeRequest`, u obfuscates either its real hidden value or any of the $k-1$ random values. Then an attacker might infer k possible answers but is not able to identify the hidden value if H is well chosen (and side channels such as topological information cannot be used to exclude certain values). For approximating the global distribution G, one subtracts the expected number of answers generated from the distribution H from the observed answers. More precisely, for samples $s_1, \ldots, s_m$, we set $P(G = x) = \frac{\max\{0,\min\{1,k\cdot\frac{|i\in\{1\ldots m\}:s_i=x|}{m}-(k-1)P(H=x)\}\}}{N}$ for the normalization factor N. Normalization and cutoff are necessary for the approximation to be a probability distribution. The choice of H is very critical, since the actual values can be identified if their probability according to H is highly different to that according to G. Nevertheless, since each node only provides one sample of G, H does not need to be a good approximation of G as long as H's support includes G's support. Besides the choice of H, the number of samples required to reliably approximate G should increase drastically.

In summary, we showed that the current obfuscation of node properties in Freenet can easily be broken using Bayesian Statistics. Our inference algorithm was found to be accurate in both simulations and measurements. Consequently, we suggested a novel obfuscation method, aiming to achieve k-anonymity. In future work, the newly suggested algorithm needs to be analyzed in depth, both with regard to the provided anonymity and its suitability for obtaining global statistics.

Appendix C

Computation of Remaining Online Time

Table 5.1 displays the remaining online time distribution R^{ind} as computed in Equation 5.14, as well as the probability that R^{ind} exceeds a constant z, i.e

$$\begin{aligned} P(R^{ind} \geq z) &= \int_z^\infty f_{R^{ind}}(x)dx \\ &= \frac{1}{\mathbb{E}(S)} \int_z^\infty 1 - F_S(x)dx. \end{aligned} \tag{C.1}$$

Now, we present the details of deriving $P(R^{ind} \geq z)$ for Weibull and lognormal distributed session lengths. Consider that as $f_{\tilde{S}}(x) = \frac{1-F_S(x)}{E(S)}$ is a density function with integral 1, we get

$$\frac{1}{\mathbb{E}(S)} \int_z^\infty (1 - F_S(x))dx = 1 - \frac{1}{\mathbb{E}(S)} \int_0^z (1 - F_S(x))dx.$$

Changing the order of integration as by Fubini's Theorem, the integral on the right side can be determined as

$$\begin{aligned} &\int_0^z (1 - F_S(x))dx \\ &= \left(z - \int_0^z \int_0^x f_S(y)dydx \right) \\ &= \left(z - \int_0^z f_S(y) \int_y^z 1dxdy \right) \\ &= \left(z - \int_0^z f_S(y)(z-y)dy \right) \\ &= \left(z \cdot (1 - F_S(1)) + \int_0^z y f_S(y)dy \right). \end{aligned} \tag{C.2}$$

Applying Equation C.2 to the Weibull distribution, we get

$$\begin{aligned} &\int_0^z (1 - F_{S^W}(z)) \\ &= z(1 - F_{S^W}(z)) + \int_0^z y f_{S^W}(y)dy \\ &= ze^{-(z/\lambda)^k} + \int_0^z k \left(\frac{y}{\lambda}\right)^k e^{-(y/\lambda)^k} dy \\ &= ze^{-(z/\lambda)^k} + \lambda \int_0^{(z/\lambda)^k} t^{1/k} e^{-t} dz \\ &= ze^{-(z/\lambda)^k} + \lambda\gamma(1 + 1/k, \left(\frac{z}{\lambda}\right)^k) \end{aligned}$$

The second last step substitutes $t = \left(\frac{y}{\lambda}\right)^k$.

In case of the Lognormal distribution, we apply the substitution $t = k\ln(y/\lambda)$ to derive the integral $\int_0^z (1 - F_{S^L}(z))$ in Equation C.2, resulting in

$$\begin{aligned}
&\int_0^z (1 - F_{S^L}(z)) \\
&= z(1 - F_{S^L}(z)) + \int_0^z y f_{S^L}(y) dy \\
&= z\left(1 - \Phi\left(k\ln\frac{z}{\lambda}\right)\right) + \int_0^z \frac{k}{\sqrt{2\pi}} e^{-\left(k\ln\frac{y}{\lambda}\right)^2/2} dy \\
&= z\left(1 - \Phi\left(k\ln\frac{z}{\lambda}\right)\right) + \int_{-\infty}^{k\ln\frac{z}{\lambda}} \frac{1}{\sqrt{2\pi}} e^{t/k+\lambda} e^{-t^2/2} dt \\
&= z\left(1 - \Phi\left(k\ln\frac{z}{\lambda}\right)\right) + \lambda e^{1/(2k^2)} \Phi\left(k\ln\frac{z}{\lambda} - 1/k\right).
\end{aligned}$$

The desired formulas for $P(R^{ind} \geq z)$ follow from Esq. C.1 and C.2, namely

$$P(R_W^{ind} \geq z) = 1 - \frac{z e^{-(z/\lambda)^k} + \lambda\gamma(1 + 1/k, \left(\frac{z}{\lambda}\right)^k)}{\lambda\Gamma(1 + 1/k)}$$

with $\mathbb{E}(S^W) = \lambda\Gamma(1 + 1/k)$ and

$$P(R_L^{ind} \geq z) = 1 - \frac{z\left(1 - \Phi\left(k(\ln z/\lambda)\right)\right) + \lambda e^{1/(2k^2)} \Phi\left(k\ln(z/\lambda) - 1/k\right)}{\lambda e^{1/(2k^2)}}$$

with $\mathbb{E}(S^L) = \lambda e^{1/(2k^2)}$.

Appendix D

Topology Aware Keys

In this chapter, we show how to assign keys to content in a topology-aware manner such that balanced content addressing can be achieved for the embedding algorithms *PrefixEmbedding* (Algorithm 7.1 in Section 7.4.2) and *PrefixSEmbedding* (Algorithm 7.3 in Section 7.4.3). We first describe our algorithms and then evaluate the algorithms with regard to the load balance in a simulation study. The simulation study indicates that indeed our topology aware keys achieve balanced content addressing at the price of slightly longer routes. The results were published as part of [139].

Note that we designed the key assignment algorithm with applications such as coordinating a single AS in content-centric networking in mind. Hence, throughout this section, we assume that the topology is i) stable, and ii) globally known.

D.1 Algorithm Design

Recall from Section 7.4 that both *PrefixEmbedding* and *PrefixSEmbedding* assign the same fraction of keys to any two leaves at the same level. Hence, for arbitrary unbalanced trees, the storage load is expected to be unbalanced, as shown in Section 7.5. In the following, we discuss how to create topology-aware keys, which achieve a uniform distribution over of keys over nodes for arbitrary topologies.

The principal idea of our key assignment algorithm is to choose the keys such that the probability that a key with prefix x has prefix $x||0$ is approximately equal to the ratio of the size of left subtree rooted at node x in the virtual tree and descendants of x.

In the following, let L denote the set of nodes that are actually assigned content, i.e., all nodes with less than 2 children for *PrefixEmbedding* and all nodes V for *PrefixSEmbedding*. Furthermore, we let $id_S(v)$ denote v's coordinate determining the fraction of stored content, i.e, it can also denote the single coordinate $id(v)$ for *PrefixEmbedding*. Our algorithm requires the use of a hash function h with image $\{0,1\}^z$ for some $z \in \mathbb{N}$ exceeding the depth of the spanning tree.

Like node coordinates in virtual binary trees, keys correspond to bit sequences $b_1 b_2 ... b_z$. We compute the key of content c iteratively, as detailed in Algorithm ComputeKey(), adding bits step-wise. Let $d_i = b_1 ... b_i$ be the assigned key after the i-th step of the key assignment for $i \geq 0$. If there is only one node v in L with d_i being a prefix of $id_S(v)$, we have found the responsible node and only add a random $z - i$ bits to the key. Otherwise, in the $i+1$-step, we derive the fraction f_1 of nodes in L whose coordinate has d_i as a prefix, i.e., is of the form $id_S(v) = d_i||postfix$, and the fraction f_2 of nodes in L whose coordinate has $d_i||0$ as a prefix. For the content addressing to be balanced, roughly f_2/f_1 of the keys with prefix d_i should have 0 as the next bit, i.e., be assigned to the left subtree rooted at the virtual node with coordinate d_i. We make use of h's pseudo-randomness by assigning the bit 0 if $h_{i+1} = h(c \oplus (i+1))/2^z \leq f_2/f_1$ and 1 otherwise. Formally, we thus derive the $i+1$-th bit of the by

$$b_{i+1} = \begin{cases} 0, \frac{h_{i+1}}{2^z} \leq \frac{|\{v \in L : cpl(id_S(v), d_i||0) = i+1\}|}{|\{v \in L : cpl(id_S(v), d) = i\}|} \\ 1, otherwise \end{cases} \tag{D.1}$$

with $cpl(x, y)$ denoting the common prefix length of two bit sequences x and y. In this manner, we subsequently assign keys such that the fraction of keys per subtree of the virtual tree roughly corresponds to the fraction of nodes that store content and are in the subtree. In particular, keys are distributed approximately uniformly on all content storing nodes.

D.2 Simulation

We evaluated our approach using 9 different topologies of autonomous systems (AS).

Algorithm D.1 computeKey(BitSequence f)

```
{Given: Graph G=(V,E), coordinates id, L: content-storing nodes}
{cpl: common prefix length, ||: concatenation}
{s[i...j]: bits i to j of s}
i = 0
key = ''
while length(key) < z do
  if |{v ∈ V : cpl(ID(v), key) = i}| = 0 then
    key = key||h(f)[i + 1, ..., z]
  else
    hash = h(f ⊕ i)/2^z
    if hash ≤ |{v∈L:cpl(id_S(v),key||0)=i+1}| / |{v∈L:cpl(id_S(v),key)=i}| then
      key = key||0
    else
      key = key||1
    end if
  end if
end while
return key
```

Metrics and Set-up We considered three metrics for our evaluation:

i) the fraction of stored items each node is responsible for,

ii) the traffic distribution, i.e., fraction of queries processed by each node,

iii) the number of hops needed to discover the destination of the query using greedy routing.

We evaluated the correlation between i) and ii) to see if a high storage load might be partly compensated by experiencing less traffic and vice versa. Ranking the nodes by i) and ii) provided an overview of how storage and traffic is balanced between the nodes. The evaluation was conducted as follows: We first created a spanning tree of the graph executing a breadth-first search starting a random node. Then we generated a set of $k = 10,000$ random ASCII character strings of length 20, which we used to represent the queried content. Afterwards, we computed the embedding for all considered embedding algorithms. For each embedding and each applicable key generation scheme, we then created the keys from the character strings and executed a query for each key from a random start node. Hence, a total of 5 combinations of embedding and key generation were evaluated : The Kleinberg embedding [87] KB with hashing into the unit disk as well as *PrefixEmbedding* P_H/P_{TA} and *PrefixSEmbedding* PS_H/PS_{TA} using both straight-forward hashing (H) and topology-aware (TA) keys. The relation between the storage and the traffic load on nodes is measured by the Pearson correlation coefficient. Results were averaged over 20 runs.

The hop count iii) does not necessarily correspond to the stretch, which is defined as the average ratio of the length of the routing path to the shortest path for all pairs of nodes. Since iii) considers queries, it is lower than the stretch if nodes that are easily discovered, e.g., the root, receive a disproportional high number of queries. The paths actually traversed during routing are more important for the working system, so that we choose this metric rather than the stretch.

Expectations We expected the Kleinberg embedding to produce a very unbalanced load distribution. The root node was responsible for the majority of the unit disk when considering Euclidean space (which the hashing did) regardless of the structure of the tree. Similarly, the tree for *PrefixEmbedding* and *PrefixSEmbedding* was bound to have leaves at very different levels, leading to a high storage load on those on a high level when using straight-forward hashing. The maximum storage load was likely to be even higher for *PrefixSEmbedding* because the virtual nodes corresponding to the storage ID of the internal nodes on the top levels are always leaves. However, for topology-aware keys the load was expected to be uniformly distributed over all nodes (*PrefixSEmbedding*) or all leaf nodes (*PrefixEmbedding*). When considering the traffic a node has to process rather than the storage load, we expected nodes on the higher levels to experience a higher load. Recall that messages between nodes did not necessarily pass their common ancestor in the tree, since greedy routing also used non-tree edges. Nevertheless, tree edges were more likely to be traversed, so that we expected an unbalanced traffic distribution for all spanning-tree based embeddings. Thus, there should also be a high positive correlation between traffic and storage

	Maximum Storage Load					Maximum Traffic				
AS	KB	P_H	P_{TA}	PS_H	PS_{TA}	KB	P_H	P_{TA}	PS_H	PS_{TA}
1221	0.952	0.410	0.081	0.401	0.011	0.970	0.804	0.515	0.829	0.648
1239	0.925	0.216	0.004	0.340	0.003	0.972	0.588	0.367	0.665	0.369
1755	0.948	0.280	0.016	0.336	0.008	0.962	0.640	0.486	0.719	0.498
2914	0.950	0.270	0.003	0.289	0.002	0.972	0.699	0.348	0.763	0.350
3257	0.950	0.337	0.010	0.375	0.006	0.975	0.813	0.562	0.841	0.574
3356	0.950	0.185	0.004	0.228	0.003	0.973	0.507	0.351	0.582	0.361
3967	0.943	0.270	0.010	0.301	0.007	0.963	0.567	0.434	0.627	0.428
6461	0.922	0.351	0.117	0.389	0.052	0.953	0.638	0.575	0.698	0.567
7018	0.927	0.238	0.004	0.286	0.003	0.973	0.597	0.401	0.648	0.379

Table D.1: Maximum load for various AS topologies with the following embedding/content addressing schemes: KB-Kleinberg Embedding, P_H-*PrefixEmbedding* with standard hashing,P_{TA}: *PrefixEmbedding* with topology-aware keys, PS_H-*PrefixSEmbedding* with standard hashing,PS_{TA}: *PrefixSEmbedding* with topology-aware keys

load for the Kleinberg and *PrefixSEmbedding* with standard hashing. Both allocated the majority of the queries and the traffic to the higher levels of the tree. When using topology-aware keys, the traffic should be uncorrelated to the uniformly distributed load for *PrefixSEmbedding*. *PrefixEmbedding* allocated all files on leave nodes. These were rarely intermediate nodes for queries, however they frequently are destinations, so that we did not know what type of correlation to expect.

Previous work on greedy embeddings showed that they exhibit similarly short routes and a low stretch [87, 56, 82]. Potentially, the average routing length is slightly lower for Kleinberg's embedding due to high fraction of queries addressed to the root node, which is fast to route to using tree edges.

Results We first consider the maximum load per node and the total traffic, for the 9 considered ASs. Afterwards, we analyze the distribution of the load for one exemplary AS, AS1239. In order to show the general applicability of our results, Table D.1 and Table D.2 summarize the maximal load and the routing length for the 9 sample ASs. In addition, the average routing length is given in order to estimate delays and the overall traffic.

For the Kleinberg embedding, the storage load was always above 90%, and the fraction of traffic the most loaded node had to process was above 95%. For *PrefixEmbedding* and *PrefixSEmbedding* with hashing, the highest storage load and traffic were 20% to 40% and 50% to 85%, respectively. The maximum load was slightly higher for *PrefixSEmbedding* because internal nodes on high levels receive a large fraction of queries. For topology-aware keys, the maximum storage load was always less than twice the average load for *PrefixSEmbedding* with the actual load depending on the size of the AS. For *PrefixEmbedding*, the maximum load was slightly higher, because only a subset of the nodes participated in storing. The maximum traffic was drastically reduced by topology-aware keys as well, being at most 35% and 65%. Here, *PrefixSEmbedding* has no overall advantage over *PrefixEmbedding*.

We also analyzed if load balancing increased the overall traffic, i.e., the number of hops needed to resolve a query. Table D.2 indicates that indeed the topology-aware keys exhibited a slightly longer routing length than with Kleinberg's embedding. However, the difference was mostly around half a hop on average and at most 0.76 hops (comparing Kleinberg KB and *PrefixSEmbedding* PS_{TA} for AS3967). Note that the difference was not due to an increased stretch, since both the standard hashing and the topology-aware keys used the same topology. Rather, the reason lay in the position of content-storing nodes in the spanning tree. *PrefixEmbedding*, storing all content at leaves, in general showed the longest routes, whereas Kleinberg embedding, storing most of the content at the root, was potentially the least costly. Due to the tree structure, the shortest path to the route was found, but non-optimal routes were common for leave nodes, so that an increased storage of content on leaves increased the routing length.

We now focus on a single AS 1239 for further analysis, but emphasize that the results applied equally to the other ASes as well.

The distribution of the storage load is displayed in Figure D.1a, using a cumulative distribution function (cdf) to show the fraction of files the k nodes with the highest load are responsible for. The curve shows a very steep initial increase for Kleinberg's embedding as well as for the two *PrefixEmbedding*s with standard hashing. By introducing topology-aware keys, the storage load was balanced uniformly, so that the increase is close to linear. The curve for *PrefixEmbedding* with topology-aware keys has a steeper slope because internal nodes with more than one child do not receive any load, whereas the load was uniformly distributed between all nodes for *PrefixSEmbedding*.

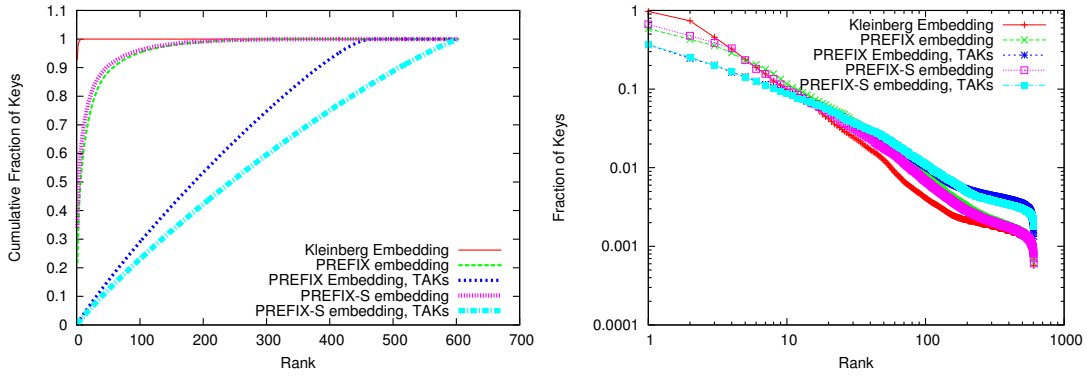
For the fraction of queries a node had to forward, i.e., the traffic per node, topology-aware keys also

AS	KB	P_H	P_{TA}	PS_H	PS_{TA}
1221	5.54	5.59	5.35	5.51	4.88
1239	4.94	5.41	5.00	5.39	5.28
1755	5.25	5.52	5.93	5.31	5.57
2914	5.90	6.22	6.36	6.01	6.20
3257	5.30	5.81	6.03	5.45	5.84
3356	3.84	4.38	4.26	4.23	4.18
3967	5.05	5.39	5.98	5.11	5.81
6461	3.34	3.55	3.57	3.42	3.41
7018	5.60	5.68	6.27	5.50	6.23

Table D.2: Routing length for various AS topologies with the following embedding/content addressing schemes: KB-Kleinberg Embedding, P_H-*PrefixEmbedding* with standard hashing,P_{TA}: *PrefixEmbedding* with topology-aware keys, PS_H-*PrefixSEmbedding* with standard hashing,PS_{TA}: *PrefixSEmbedding* with topology-aware keys

lessened the imbalance, but could not abolish it (see Figure D.1b). The nodes with the highest load were involved in more 35 % of all queries. However, the traffic load for topology-aware keys was considerably less than for hyperbolic embeddings, for which the root node is involved in more than 98 % of the queries.

As can be expected from these results, the correlation coefficients of the storage and the traffic load were high for Kleinberg (0.704) and *PrefixSEmbedding* (0.629), whereas there was no notable correlation for *PrefixSEmbedding* with topology-aware keys (0.011). For *PrefixEmbedding* was correlation coefficient is clearly positive (0.316) for straight-forward hashing and clearly negative (−0.348) for topology-aware keys. The result was due to the fact that leaves nodes at a high level were frequent destinations of queries while at the same time store a large number of items for the standard addressing scheme, leading to a positive correlation. For topology-aware keys, items were still stored only on leaf nodes, but uniformly distributed between them. Hence none of them had a dis-proportionally large amount of traffic, which is reserved for the internal nodes without storage responsibility, leading to a negative correlation.



(a) Storage (b) Traffic

Figure D.1: Load distribution in AS topologies: a) CDF of storage by rank, b) fraction of traffic ranked

We have shown that the poor load balance of embeddings can be improved. Topology-aware keys achieve a uniform storage load and reduce the traffic at congested nodes at the price of a marginally increased overall traffic. The above results indicate that *PrefixSEmbedding* is the best choice when combined with topology-aware keys, since it offers a uniform storage distribution over all nodes and the lowest maximum traffic. However, *PrefixEmbedding* offers a negative correlation between storage and traffic, so that congested nodes only have to forward queries rather than answer them. Depending on the actual scenario, in particular storage and time constraints, *PrefixEmbedding* can be a better choice.

D.3 Discussion

While topology-aware keys present a valid approach for realizing balanced content addressing in stable networks with a globally known structure, they are not an option for F2F overlays. Apart from revealing potentially critical information, the constant dynamics imply a constant change of the topology-aware key of a file. Such inconsistency with regard to the key assignment is bound to result in lookup failures

and high overhead. While the dynamic assignment of keys can be substituted with a dynamic coordinate assignment, the need for topology information and thus a loss of attack resilience is inherent to the tree structure. In summary, the presented algorithm is highly promising for content-centric networks but of little to no relevance for F2F overlays.

Appendix E

PPP Return Addresses

In this section, we show how to obfuscate return addresses in VOUTE further. However, the presented return addresses result in a higher routing complexity. Our *potential path preserving (PPP)* return addresses allow a node v to detect which neighbor is closer to the destination but prevent v from identifying the closest neighbor. Thus, greedy paths are still identified as such but it is unclear which of multiple greedy path is chosen. By choosing a random closer neighbor rather than the closet neighbor, the expected routing length is increased.

In this section, we only explain the address generation algorithm $\mathbf{AdGen}_{node}^{PPP}$. Our simulation-based evaluation is presented as part of our additional results in Chapter F.

PPP addresses are generated by adding a layer of symmetric encryption to RAP return addresses generated by Algorithm 8.3. The idea of the approach is to allow u to determine if the common prefix length of a coordinate is longer than $cpl(cord(y), id(u))$ but not the actual length. For this reason, the additional layer can only be applied when using the common prefix length-based distance δ_{CPL}. In a nutshell, we generate PPP return address through symmetric encryption of a RAP return address using key material only known within certain subtrees.

Let $Enc : H \times \mathbb{K}_{Sym} \to H$ be a semantically secure symmetric encryption function onto h's image H with keyspace $\mathbb{K}_{Sym}$. $Dec : H \times Sym \to H$ denotes the corresponding decryption. For each subtree of the spanning tree, we distribute keys. During the key distribution, each internal node w at level l generates a symmetric key $k_l(w)$ by a pseudo-random key generation algorithm $SymGen$. Subsequently, w distributes $k_l(w)$ to all its descendants. In this manner, a node v at level $\tilde{l}$ obtains keys $k_1(v), \ldots, k_{\tilde{l}-1}(v)$ such that $k_\lambda(v)$ was generated by v's ancestor at level λ and forwarded to v along the tree edges. So, $k_\lambda(v)$ is known to all nodes having a common prefix length of at least λ with v. After generating a RAP return address $y = (d_1, \ldots, d_L)$, v additionally encrypts the $\lambda + 1$-th element with the key $k_\lambda(v)$, constructing the return address $y' = (d'_1, \ldots, d'_L)$ with

$$d'_j = \begin{cases} Enc(k_{j-1}(v), d_j), & 2 \leq j \leq l \\ d_j, & \text{otherwise} \end{cases}. \tag{E.1}$$

The second case in Equation E.1 treats the first element, which remains unencrypted, and the randomly chosen padding. After generating y', v publishes y', the routing information $\tilde{k}$ for generating y and $mac(\mathbb{K}_{MAC}(v), y')$. The pseudo code of the additional encryption is displayed in Algorithm E.1.

A third realization $\mathbf{R}^{PPP}$ of the routing algorithm $\mathbf{R}_{node}$ is given by the construction of PPP addresses. During routing, a node u at level l first applies the decryption function to the second to $l+1$-th element of the return address $y' = (d'_1, \ldots, d'_L)$. So, v obtains $f(y') = (z_1, \ldots, z_{l+1})$ with

$$z_j = \begin{cases} d'_1, & j = 1 \\ Dec(k_{j-1}(u), d'_j), & \text{otherwise} \end{cases}.$$

Afterwards, u determines $cpl(f(y'), cash(\tilde{k}, c))$ for all coordinates c in its neighborhood. Note that $cpl(f(y'), cash(\tilde{k}, c)$ is only a lower bound on $cpl(cord(y), c) = cpl(y, cash(\tilde{k}, c)$ because u is not able to correctly decrypt some elements of y'. Based on the common prefix length, v can evaluate the diversity measure

$$\delta^u_{PPP-CPL}(y', \tilde{k}, c) = \delta_{CPL}(f(y'), cash(c, \tilde{k})) \tag{E.2}$$

for the distance δ_{CPL} defined in Equation 8.2. In this manner, the node u obtains a set of all neighbors closer to the destination than itself. So, u chooses a random node from this set as the next hop.

We here give a short intuition on why Algorithm E.1 indeed only reveal if the common prefix length of a neighbor is longer but does not reveal the actual length of the common prefix. Let u be a node and y' be a return address generated by v, a node at level l_v. If $cpl(id(v), id(u)) = \lambda$, u correctly decrypts the first $\lambda + 1$ elements of y' because $k_i(u) = k_i(v)$ for $i = 1 \dots \lambda$. Due to the semantic security of the symmetric encryption, u cannot infer information about the remaining elements of y from $d'_{\lambda+2}, \dots, d'_{l_u}$ because u does not know $k_i(v)$ for $i > \lambda+1$. Thus, y' indeed only reveals if a coordinate c shares a longer common prefix with $cord(y)$ than $id(v)$.

Algorithm E.1 addPPPLayer()

{*Input: RAP return address* $y = (d_1, \dots, d_L)$}
{*Internal State: Keys* $k_1(v), \dots, k_{l-1}(v)$, Enc_{Sym}}
1: **for** $i = 2 \dots l$ **do**
2: $d_j \leftarrow Enc_{Sym}(k_{j-1}(v), d_j)$ {*Encrypt element* j}
3: **end for**

Appendix F

Extended Results for VOUTE

In this section, we present additional results for VOUTE, as introduced in Chapter 8. The additional results are mostly concerned with a comparison of the results for the Facebook graph *FB* to the web-of-trus *WOT* and the special-purpose social network *SPI*, introduced in Section 4.2. We compared the graphs with regard to both average routing length as well as the resilience to attacks. In addition to the comparison, we considered the impact of additional parameters and the distribution of the traffic on nodes.

All results in this section are averaged over 20 with either $10,000$ (Sections F.2 and F.4) or $100,000$ (Sections F.1 and F.3) requests each and presented with 95% confidence intervals.

F.1 Routing length of *WOT* and *SPI*

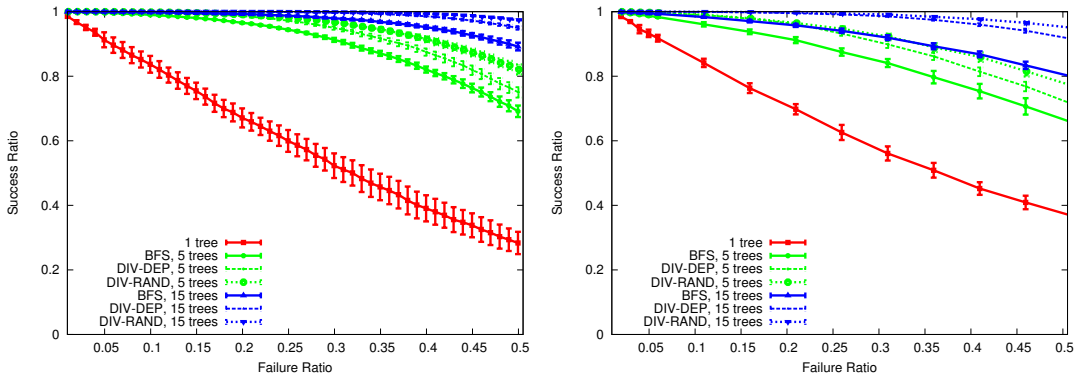
We evaluated the routing length, i.e., the number of hops one the shortest path found in any γ parallel embeddings, for all three graphs *FB*, *SPI*, and *WOT*. Our simulation set-up and notation was analogously to Section 8.3.2. In particular, we evaluated three different tree construction algorithms and three diversity measures. The three tree construction algorithms were i) breadth-first search (*BFS*), ii) Algorithm 8.1 with $q = 0.5$ and a depth-dependent parent choice in the presence of ties (*DIV-DEP*), and iii) Algorithm 8.1 with $q = 0.5$ and a random parent choice in the presence of ties (*DIV-RAND*). The three diversity metrics were i) *TD*, the tree distance for RAP return addresses, ii) *CPL*, the common prefix length-based distance for RAP return addresses, and iii) *PPP*, the common prefix length-based distance for PPP return addresses (See Section 8.2 and Appendix E for a formal definition). For brevity, we focused on $\gamma = 1, 5, 15$ trees. Note that for one tree Algorithm 8.1 results in a breadth-first search regardless of the parameters as only one parent per node is selected.

		FB			*SPI*			*WOT*		
γ	$\mathcal{TC}$	*TD*	*CPL*	*PPP*	*TD*	*CPL*	*PPP*	*TD*	*CPL*	*PPP*
1	*BFS*	6.30	6.31	6.39	6.70	6.70	6.78	6.35	6.36	6.54
5	*BFS*	5.09	5.09	5.18	5.37	5.37	5.45	5.32	5.32	5.40
	DIV-DEP	5.21	5.22	5.33	5.48	5.48	5.59	5.39	5.39	5.55
	DIV-RAND	5.54	5.56	5.74	5.66	5.69	5.86	5.49	5.50	5.73
15	*BFS*	4.67	4.67	4.74	4.90	4.90	4.95	5.10	5.10	5.14
	DIV-DEP	4.80	4.80	4.91	5.01	5.02	5.11	5.16	5.17	5.30
	DIV-RAND	5.21	5.25	5.49	5.26	5.29	5.51	5.30	5.31	5.62

Table F.1: Comparison of the routing length for different tree construction algorithms $\mathcal{TC}$ and three different diversity measures *TD*, *CPL*, *PPP* for our three exemplary real-world social networks. For $\gamma = 1$, all algorithms $\mathcal{TC}$ result in a breadth-first search.

Before detailing the results, recall the results on the shortest path length in Table 4.2. All three networks exhibit an average shortest path length between 4 and slightly above 5 with paths in *FB* being shorter than in *SPI* and *WOT*. We expected this relation to prevail for the length of the discovered routes, as the routes in the embedding have been observed to be close to the shortest path.

In the following, we present the results concerning the comparison of the social graphs rather than the different parameter settings. The latter has been considered in detail for *FB* in Section 8.3.2. We summarized the results on the average routing length in Table F.1. As expected, the routes in *FB* were the shortest for all considered parameters, varying between 4.67 and 6.39. In contrast, routes in *SPI* were only shorter than routes in *WOT* if routing utilized a high number of parallel embeddings, i.e.,



(a) *FB* (b) *WOT*

Figure F.1: Comparing the robustness to failures for the Facebook topology *FB* and the web-of-trust *WOT*: *WOT* offers fewer connections so that additional trees do not improve the fraction of available routes as strongly as for *FB*

$\gamma = 15$ in Table F.1. The reason for this divergence in relation to the shortest path length was the degree distribution. *WOT* contains a large number of nodes with a very low degree, having a median degree of only 4. For those nodes, few neighbors were neither parent nor child in the one spanning tree. Thus, the routes in the tree were closer to the shortest paths than for *SPI*. As a consequence, a higher number of trees was required to include the majority of short paths for *SPI* than for *WOT*, resulting in the expected shortest routes for *SPI* and $\gamma = 15$. All in all, the discovered routes were close to the shortest paths for all three graphs.

F.2 Robustness and Censorship-Resistance of *WOT* and *SPI*

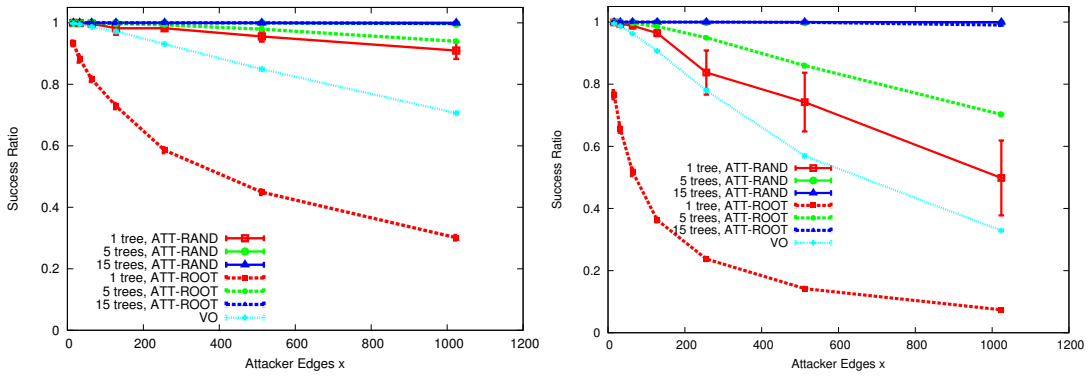
As in Section 8.4.3, we compared the robustness and resistance of embedding-based routing for $\gamma = 1, 5, 15$ to the virtual overlay *VO*. Recall that *VO* constructed a virtual Kademlia overlay on top of the F2F overlay, such that each connection in the virtual overlay corresponds to a shortest path, called tunnel, in the F2F overlay. Routing between virtual neighbors failed if their tunnel contained one or more failed or malicious nodes (but due to backtracking routing in the virtual overlay could still take alternative path).

We utilized the same parameter setting as in Section F.1. When evaluating the robustness, we removed nodes in steps of 1% up to 5% and then in steps of 5% up to 50%. In case of the censorship-resistance, we considered the two attack strategies *ATT-ROOT* and *ATT-ROOT* like in Section 8.4. Furthermore, we again focused on the tree construction *DIV-DEP* with $q = 0.5$, i.e., nodes accept an non-optimal parent with probability 0.5 and prefer parents close to the root for the resistance. Throughout this section, we focus on the diversity measure *CPL*.

We expected a slightly lower robustness and censorship-resistance for *WOT* and *SPI* than for *FB*, because the two networks are less densely connected (average degrees of below 14 and 11 rather than above 25) and are hence less likely to offer alternative routes. The lack of alternative routes should be more noticeable for multiple trees, since the lower number of potential parents entails similar trees. Thus, further trees offer fewer additional routes. However, we nevertheless expected the relation between the different algorithms and parametrization to be similar for all three topologies.

Our results agreed with our expectations. Figure F.1 displays our results with regard to the robustness to failures for *FB* and *WOT*. When routing based on one tree. *WOT* was actually more robust than *FB*, exhibiting a success ratio of close to 40% despite half the nodes failing. The high robustness was due to the few nodes with a very high degree. As long as they were online, they provided routes for most of the remaining nodes. However, *FB* achieved a higher success ratio for multiple trees. If $\gamma = 5$, *FB* had a success ratio above 85%, while *WOT*'s success ratio varied between 80 and 90%. If $\gamma = 15$, *FB*'s success ratio increased to above 95%, while *WOT*'s success ratio is only around 90% for some parameters. As expected, the higher connectivity of *FB* entailed a higher success ratio for multiple trees.

Now, we compare the three topologies with regard to their censorship-resistance considering both attackers manipulating the root election (*ATT-ROOT*) or only faking the prefix of the children (*ATT-RAND*). When comparing *FB* and *WOT*, the results for the robustness were equally valid for the resistance (see Figures 8.8b and F.2a for a detailed comparison). Comparing *WOT* and *SPI* showed that the attack resistance of *SPI* was lower for all considered parameters, as displayed in Figure F.2. The high impact of the attacks was potentially due to the small size of *SPI* with less than $10k$ nodes in comparison



(a) *WOT* (b) *SPI*

Figure F.2: Comparing the resistance of web-of-trust topology *WOT* with roughly $40k$ nodes to special-purpose social network *SPI* with less than $10k$ nodes: *SPI*'s resistance is generally lower but the relation between the different approaches is similar for both topologies (as for *FB*, see Figure 8.8b); parametrization: Distance *CPL* and tree construction *DIV-DEP* for $\gamma = 1, 5, 15$; attacks consider secure (*ATT-RAND*) and insecure root election (*ATT-ROOT*); comparison to Kademlia-based virtual overlay *VO*

to $40k$. In particular, $x = 1024$ attacker edges imply that more than 10% of all nodes had a compromised neighbor. Despite such a high fraction of edges to malicious nodes, the resistance of *SPI* was high for multiple trees with very few failed requests for $\gamma = 15$.

All in all, the results strengthens the conclusion from Section 8.4 that our approach is highly resilient, even in case of less densely connected overlays.

F.3 Traffic per Node

In this thesis, we considered the load of a node in terms of the number of stored files. However, the traffic per node, i.e., number of forwarded requests, is of interest as well. If nodes such as the root have to forward a large fraction of requests, requests might be dropped due to insufficient bandwidth. Without a concrete load model defining the request frequency and payload, it is hard to analyze whether our approach distributes the traffic adequately. The question is nevertheless extremely pressing due to the hierarchical structure of trees, which is bound to result in a more traffic for nodes close to the root. While we did not focus on this metric throughout the thesis, we present a short evaluation here.

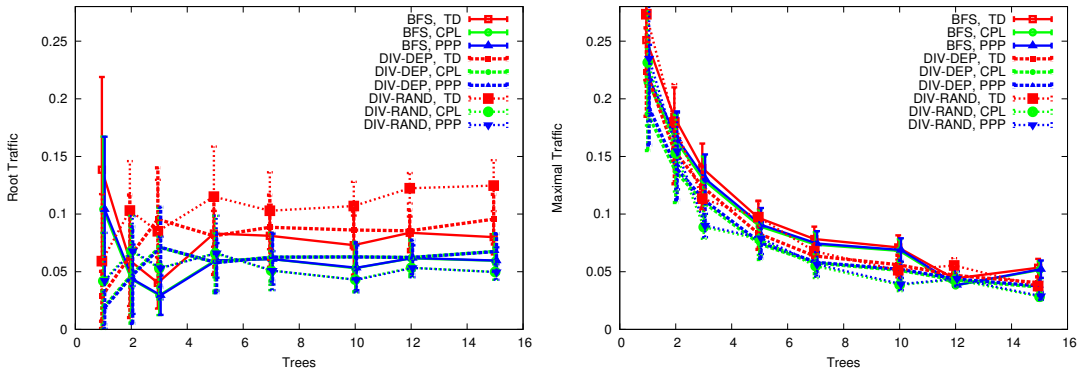
We evaluated the traffic on nodes in parallel to the efficiency evaluation presented in Section 8.3.2 for *FB* and hence used the same set-up. So, we varied the number of trees $\gamma = 1, 2, 3, 5, 7, 10, 12, 15$ and used the diversity measures *TD*, *CPL*, and *PPP* for determining the next hop on the route. Furthermore, we compared three tree construction schemes *BFS*, *DIV-DEP*, and *DIV-RAND*. In order to compare different numbers of trees, we routed in $\tau = 1$ randomly selected tree of γ trees for a source-destination pair. We measured the fraction of requests that passed the root of the selected tree as well as the maximal fraction of requests forwarded by any node in the overlay, referred to as *maximal traffic*. We compared our results to Freenet and *VO* using a degree of parallelism $\alpha = 1$ and $\alpha = 3$ for the latter.

As stated above, we expected a high fraction of requests to be forwarded via roots. Note that this fraction should not change with number of trees. However, with multiple trees, the root traffic is divided upon several nodes, so that we expected the maximal traffic to decrease with the number of trees. Furthermore, the root traffic should be higher for the tree distance *TD* as it prefers nodes close to root. In contrast to tree-based embeddings, Freenet and *VO* do not rely on hierarchical structures and thus should not exhibit a great imbalance in the traffic but a potential preference for high-degree nodes. Nevertheless, we have seen in Section 8.3.2 that tree-based embeddings exhibited much shorter routes, meaning that the average traffic per node was considerable reduced in comparison to Freenet and *VO*. When distributing the load on multiple trees, we thus expected the maximal traffic to be at least comparable and maybe even lower for our scheme in relation to the state-of-the-art.

Our results, displayed in Figure F.3, validated our expectations. Indeed, a large fraction of requests, namely roughly 10%, was forwarded via the root of the selected tree, independently of the number of trees, and the fraction was higher for *TD* than for the two remaining diversity measures. By distributing the traffic on multiple trees and thus using different root nodes, the maximal traffic fell from close to 30% to 5%. Note that the high variance indicates that the maximal traffic was highly dependent on the tree

structure, varying greatly with the structure of the spanning trees and the degree of the roots. Indeed, the shorter routes in our scheme counteract the preference of high level nodes, so that our scheme usually achieved a lower maximal traffic than the related work with roughly 80% (Freenet), 15% (VO, $\alpha = 1$) and 34% (VO, $\alpha = 3$).

The comparison of the maximal traffic with state-of-the-art approaches is highly promising, indicating that our use of trees does not necessarily create a greater imbalance. In addition, the evaluation reveals that the maximal traffic depends on the tree construction, so that future work should focus on identifying tree structures providing a good balance.

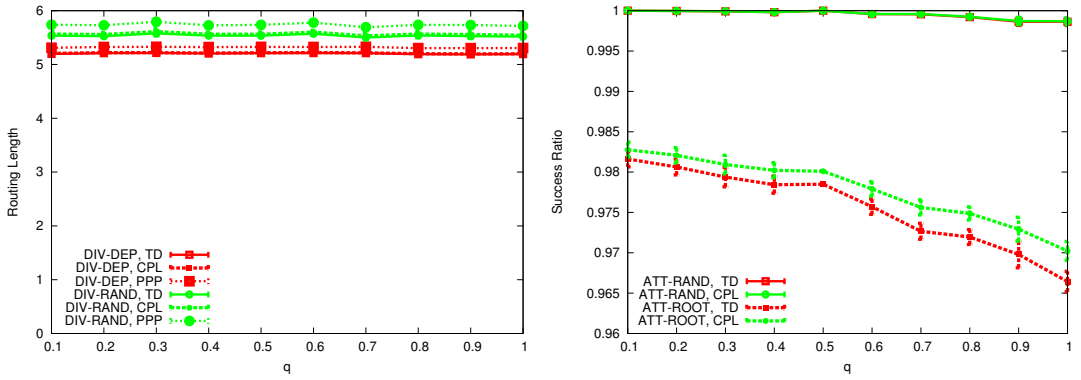


(a) Root Traffic (b) Maximal Traffic

Figure F.3: a) fraction of requests routed via roots and b) maximal fraction of requests routed via a node, choosing one of γ trees randomly to deliver the request

F.4 Impact of q

In Chapter 8, we introduced the parameter q for the construction of multiple spanning trees, which represents the probability to select a parent in multiple trees despite the existing of additional neighbors. Yet, we have not evaluated the impact of q. Instead, we focused on $q = 0.5$. Now, we consider the routing length and censorship-resistance for varying q.



(a) Routing Length (b) Resistance

Figure F.4: Impact of parameter q, the probability to accept a parent if none of the preferred neighbors offers an invitation when establishing $\gamma = 5$ trees on a) routing length for diversity metrics *TD*, *CPL*, *PPP* and either random parent selection (*DIV-RAND*) or depth-dependent selection (*DIV-DEP*) and b) censorship-resistance for diversity metric *CPL*, tree construction *DIV-DEP*, and $x = 1024$ attacker edges

More precisely, we considered the graph *FB* and varied q between 0.1 to 1.0 in steps of 0.1. We fixed the number of trees to be $\gamma = 5$ and used both depth-dependent (*DIV-DEP*) and random parent choice (*DIV-RAND*). Furthermore, we utilized the diversity measures *TD*, *CPL*, and *PPP*. For the censorship-resistance, we focused on the scenario of $x = 1024$ attacker edges and the tree construction *DIV-DEP*. Both attack strategies *ATT-RAND* and *ATT-ROOT* were evaluated.

Given the small difference between *BFS* and *DIV-DEP* with regard to efficiency in Section 8.3.2, we did not expect q to drastically impact the routing length. In contrast, a higher q implies a higher chance to accept the same parent in multiple trees, so that we expected a lower resistance for high q.

Indeed, we did not measure any significant impact of q on the routing length, as displayed in Figure F.4a. In contrast, we saw a slight decrease in resistance with q. As displayed in Figure F.4b, the success ratio was essentially 1 for $q = 0.1$ but then dropped to slightly but noticeably below 1 for *ATT-RAND*. For *ATT-ROOT*, the failure ratio was increased by roughly factor 2, from less than 2%, i.e., a success ratio of above 98%, to close to 4% for both *TD* and *CPL*. Still, the success ratio was higher than for the state-of-the-art approach regardless of the choice of q (see Figure 8.8b for comparison).