Network Architectures
and Services
NET 2011-01-1

Dissertation

Resilient Application Layer Signaling based on Supervised Peer-to-Peer (P2P) Networks

Ali Fessi

Network Architectures and Services
Department of Computer Science
Technische Universität München

Technische Universität München

Lehrstuhl für Netzarchitekturen und Netzdienste

Resilient Application Layer Signaling based on Supervised Peer-to-Peer (P2P) Networks

Ali Fessi

Vollständiger Abdruck der von der Fakultät für Informatik
der Technischen Universität München
zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Thomas Neumann
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Georg Carle
	2.	Prof. Dr. Radu State (Université de Nancy 1, Frankreich)

Die Dissertation wurde am 06.09.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.12.2010 angenommen.

Cataloging-in-Publication Data
Ali Fessi
Resilient Application Layer Signaling based on Supervised Peer-to-Peer (P2P) Networks
Dissertation, December 2010
Network Architectures and Services, Department of Computer Science
Technischen Universität München

ISBN 3-937201-17-3
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
Network Architectures and Services NET-2011-01-1
Series Editor: Georg Carle, Technischen Universität München, Germany

Abstract:

The Internet has become a critical infrastructure which requires adequate protection from errors and attacks. It plays a critical role in delivering services, e.g., human and device communication, disseminating news and information, power grid control etc. Voice over IP (VoIP) is one of the critical services which require increased resilience on the Internet. It has been widely adopted by telephony providers to reduce costs. However, many incidents have demonstrated the lack of reliability inherent in VoIP.

This thesis investigates the benefits of using *supervised* Peer-to-Peer (P2P) networks for resilient application layer signaling, specifically for VoIP session setup. The *supervisor* provides the peers with verifiable identities, thus mitigating critical security issues in P2P networks, e.g., Sybil attacks. The supervisor is involved in the signaling for VoIP session setup under normal operation. However, in case the supervisor becomes unreachable, the VoIP service can still be provided by the P2P network. A supervised P2P network can be considered as a solution between server-based and pure P2P-based signaling solutions. The goal is the combination of the advantages of both architectures leading to improved reliability and security. A reliability analysis of the supervised approach is provided based on reliability theory and traces from the Skype network. Furthermore, different mechanisms are discussed how the approach can optimize the operation of the P2P network, notably load balancing and routing.

A security threat analysis evaluates to what extent the supervised P2P approach is able to provide the same security level as in server-based signaling solutions. Finally, storing user contact data in a P2P network raises also privacy issues. Thus, concepts and experiences learned from anonymization networks such as Tor and I2P are adjusted to build a privacy preserving P2P signaling protocol for VoIP.

The theoretical analysis in this thesis has been accompanied by implementation work and experimentation on PlanetLab.

Publications:

Parts of this thesis have been pre-published in earlier versions:

- A. Fessi, H. Niedermayer, H. Kinkelin, and G. Carle. A Cooperative SIP Infrastructure for Highly Reliable Telecommunication Services. In *IPTComm '07: Proceedings of the 1st international conference on Principles, Systems and Applications of IP Telecommunications*, New York, NY, USA, 2007. ACM.

- A. Fessi, H. Niedermayer, H. Kinkelin, and G. Carle. CoSIP — a Hybrid Architecture for Reliable and Secure SIP Services. *PIK – Praxis der Informationsverarbeitung und Kommunikation*, 30(4), 2007.

- N. Kammenhuber, A. Fessi, and G. Carle. Resilience: Widerstandsfähigkeit des Internets gegen Störungen - Stand der Forschung und Entwicklung. *Informatik-Spektrum (Special Issue on Future Internet)*, 33, Number 2:131–142, April 2010.

- A. Fessi, N. Evans, H. Niedermayer, and R. Holz. Pr^2-P2PSIP: Privacy Preserving P2P Signaling for VoIP and IM. In *IPTComm '10: Proceedings of the 1st international conference on Principles, Systems and Applications of IP Telecommunications*, Munich, Germany, 2010. ACM.

CONTENTS

Part I Introduction and Background 11

1. Introduction . . . 13
1.1 Outline and Scientific Contributions . . . 15

2. Resilience of Internet Services . . . 19
2.1 Terminology . . . 19
2.1.1 The 'fault → error → failure' chain . . . 20
2.1.2 Fault Tolerance . . . 20
2.1.3 Resilience . . . 20
2.1.4 Dependability . . . 21
2.1.5 Survivability . . . 23
2.1.6 Security . . . 23
2.1.7 Relationship between Security, Dependability and Resilience . . . 24
2.1.8 Conclusions . . . 25
2.2 Challenges . . . 25
2.2.1 Topological Failures . . . 25
2.2.2 Overload . . . 26
2.2.3 Lack of Integrity . . . 26
2.2.4 Software Faults . . . 26
2.2.5 Correlated and Cascaded Failures . . . 27
2.3 Mechanisms . . . 27
2.3.1 Protection at the Application Layer . . . 27
2.3.2 Protection at the Lower Layers . . . 31
2.4 Reading Recommendations . . . 33
2.5 Conclusions . . . 33

3. Overlay and P2P Networks . . . 35
3.1 Terminology . . . 35
3.1.1 P2P networks . . . 35
3.1.2 Overlay Networks . . . 35
3.1.3 Unstructured Overlays . . . 36
3.1.4 Structured Overlays . . . 36
3.1.5 Distributed Hash Tables (DHTs) . . . 36

3.2 Algorithmic Aspects . . . 36
3.2.1 ID Space . . . 36
3.2.2 Distance Metric . . . 37
3.2.3 Consistent Hashing . . . 37
3.2.4 Geometry . . . 38
3.2.5 Routing . . . 41
3.3 Operational Aspects . . . 41
3.3.1 Bootstrapping . . . 41
3.3.2 Maintenance . . . 42
3.3.3 Parallel Lookups . . . 42
3.3.4 Recursive vs. Iterative Routing . . . 43
3.3.5 Middlebox Traversal . . . 43
3.3.6 Underlay Proximity . . . 43
3.3.7 Resource awareness . . . 44
3.3.8 Security . . . 44
3.4 Conclusions . . . 44

Part II P2P Networks and Applications for Resilient Internet Services 45

4. Coooperative SIP (CoSIP) . . . 47
4.1 Background . . . 47
4.1.1 Session Initiation Protocol (SIP) . . . 47
4.1.2 Challenges in VoIP/SIP Networks . . . 49
4.1.3 P2P-based SIP (P2PSIP) . . . 50
4.2 CoSIP Overview . . . 51
4.2.1 Concept . . . 51
4.2.2 Application Scenarios . . . 53
4.3 Prototype Implementation . . . 54
4.4 Evaluation . . . 55
4.4.1 Reliability Analysis . . . 55
4.4.2 Security . . . 63
4.4.3 Evaluation on PlanetLab . . . 64
4.5 Related Work . . . 66
4.6 Conclusions . . . 68

5. Supervised Node IDs . . . 71
5.1 Background . . . 72
5.2 Near-Optimal Node IDs . . . 72
5.3 Load Distribution . . . 74
5.3.1 Simulations . . . 74
5.3.2 Mathematical Model . . . 74

5.4 Maximum Load Distribution . . . 82
5.4.1 Simulation Setup . . . 83
5.4.2 Interpretation . . . 84
5.5 Related Work . . . 86
5.6 Conclusions . . . 88

6. Low-Diameter Overlays . . . 91
6.1 Background . . . 91
6.1.1 Optimal-Diameter Graphs . . . 91
6.1.2 Hypercubes . . . 92
6.1.3 Generalized Hypercubes . . . 92
6.2 Overlay Algorithm . . . 94
6.2.1 Network Parameters and Notation . . . 94
6.2.2 Overlay Routing Tables . . . 94
6.2.3 Routing Algorithm . . . 96
6.3 Evaluation . . . 98
6.3.1 Mathematical Model . . . 99
6.3.2 Simulations . . . 102
6.4 Supervised Low-Diameter Overlays . . . 103
6.4.1 Approach . . . 103
6.4.2 Evaluation . . . 106
6.5 Related Work . . . 106
6.6 Conclusions . . . 107

Part III Security and Privacy 109

7. CoSIP Security . . . 111
7.1 Security Requirements . . . 111
7.2 Threat Model . . . 113
7.3 Mechanisms . . . 114
7.3.1 Secure Node ID Assignment . . . 114
7.3.2 Resilient Routing . . . 116
7.3.3 Data Replication . . . 119
7.4 Evaluation . . . 120
7.5 Related Work . . . 120
7.6 Conclusions . . . 121

8. Privacy-Preserving P2PSIP . . . 123
8.1 Design of Pr^2-P2PSIP . . . 123
8.1.1 Model and Notation . . . 124
8.1.2 Protocol Overview . . . 127
8.1.3 Protocol Operations . . . 129

8.1.4 Cryptographic Primitives . . . 133
8.2 Evaluating Pr^2-P2PSIP . . . 134
8.2.1 Threat Analysis . . . 134
8.2.2 Reliability Cost Analysis . . . 140
8.2.3 Cryptographic Overhead . . . 144
8.2.4 End-to-end Signaling Latency . . . 145
8.3 Related Work . . . 145
8.4 Conclusions . . . 147

Part IV Conclusions and Outlook 149

9. Conclusions and Outlook . . . 151
9.1 Summary . . . 151
9.2 Future Directions . . . 153

Appendix 157

A. CoSIP Prototype Implementation Details . . . 159
A.1 Design Decisions . . . 159
A.1.1 Support of Different SIP Clients Software . . . 159
A.1.2 Choice of a DHT . . . 159
A.1.3 A Supernode Approach . . . 160
A.1.4 Putting Everything Together . . . 160
A.1.5 Support of P2P-based SIP . . . 160
A.2 Message Processing . . . 161
A.2.1 Session Establishment with CoSIP . . . 161
A.3 High-Level State Machines . . . 162
A.3.1 `REGISTER` Session State Machine . . . 162
A.3.2 `INVITE` Session State Machine . . . 163

B. SIP-Based X.509 Certificate Enrollment . . . 167

C. Taxonomy of Attacks on P2P Networks . . . 171
C.1 Attacks on the Overlay . . . 171
C.2 Attacks on the DHT . . . 176
C.3 Attacks on the Application . . . 178

Bibliography 181

Part I

INTRODUCTION AND BACKGROUND

1. INTRODUCTION

Communication has always been a fundamental need for human beings to exchange knowledge, experience and emotions. The Internet is a communication medium which has changed the way we live. From a technical point of view, there are some good reasons for the success of the Internet. Leonard Kleinrock worked in the 1960s on the mathematical model behind packet switching [71]. He demonstrated that packet switching provides better bandwidth utilization and response time than the traditional circuit switching technology used for telephony.

Meanwhile, the Internet has dominated other communication technologies such as Asynchronous Transfer Mode (ATM), Public Switched Telephone Network (PSTN) and Integrated Services Digital Network (ISDN). The television network is one of the next candidates to be replaced by IP-based TV. The usage of IP also is being investigated within airplanes [24]. The Internet has increasingly been used for the control of the power grid network [116]. These examples demonstrate how the Internet has become a *critical infrastructure* on which our life and prosperity depends. The success of another communication technology, GSM (Global System for Mobile Communications: originally from Groupe Spécial Mobile), lead to the extensive exploration of Internet support in 3rd generation (3G) mobile networks. 3G networks consist of both circuit-switched and packet-switched domains. However, the trend is to discard the circuit-switched domain for the benefit of the packet-switched domain in the successors, the 3GPP Long Term Evolution (LTE) networks [55]. The circuit-switched domain is not required for telephony anymore given that voice data will be transported exclusively using Voice over IP (VoIP).

The Lack of Resilience in the Current Internet

The pervasive use of the Internet today notably for critical applications increases the resilience requirements on the net. At the early time of the Internet, hosts were stationary, users had a strong technical background and were trustworthy. Applications were limited to web browsing and emails. Today, the Internet user base has grown to nearly two billion users [105] with more than 35,000 registered autonomous systems (ASs) [14]. Even if equipment or protocols are extensively tested during the development phase, it can not be guaranteed that they can provide the required functionality

under real world conditions, particularly since the deployment conditions evolve continuously over time, e.g., traffic properties, user base, applications and physical or environmental conditions. Examples of physical or environmental conditions are cable damages, power failures or interactions with other protocols and equipments.

Jean-Claude Laprie defines 'resilience in computer systems' as

- *the persistence of the avoidance of failures that are unacceptably frequent or severe, when facing changes* [74].

Given that 'resilience' includes the aspect of *changes*, guaranteeing continuous resilience is a very challenging task, most notably because changes are not always predictable. Changes can be triggered by non-malicious or malicious actions. Thus, resilience includes security aspects as well.

Peer-to-Peer (P2P) Networks as Resilience Enablers

A networking paradigm which garnered much attention since the end of the 1990's offer attractive properties from a resilience perspective: *P2P networking*. P2P networks are generally *overlays* on top of existing networks, which allow for functionality not provided by the underlay, e.g., multicast or distributed lookup. In contrast to client/server systems with asymmetric roles, each peer in a P2P network implements the functions of both client and server.

P2P networks are *decentralized* by design. They are built to cope with churn, i.e., with peers joining and leaving the network regularly. Also, data stored in a P2P network is *replicated* at multiple nodes in the network. Nodes can detect that their neighboring peers have left the network and thus, can *autonomously recover from stale routing table entries*. Moreover, P2P networks offer *geographic diversity*. Inherent decentralization, data replication, autonomous recovery from stale routing table entries, and geographic diversity, these are properties which make P2P networks attractive for building resilient network services.

The Lack of Security in Pure P2P Networks

On the other hand, building services using P2P networks is not straightforward. Given that the service is provided by the peers, it depends on the trustworthiness of the peers whether they perform their tasks, notably storage and routing, correctly and cooperate to provide a resilient service. If a P2P network is open to the public, which is the case for many file sharing networks, e.g., KAD [140], then a number of attacks are possible, particularly Sybil [37] attacks where an attacker emulates a large number of peers in the network, and eclipse [135] attacks where attackers hide content or nodes in the network and prevent them from being reachable. These attacks can render the aforementioned resilience benefits of P2P networks invalid.

Supervised P2P Networks

To address the security issues in P2P networks and benefit appropriately from their resilience advantages, we follow an approach with a *supervised* P2P network in this thesis.

Definition A *Supervised P2P network* is a P2P network with a node with a special role: the P2P network supervisor. The supervisor is involved in building the P2P network but is not required during the normal network operation. Thus, the failure of the supervisor does not lead to the failure of the service offered by the P2P network.

Supervised P2P networks have been under-explored, especially from a security perspective. The P2P research community has invested considerable effort into addressing P2P security issues in a distributed manner [11,31,33, 34,83,150]. However, these solutions can not provide resilience guarantees. One of the main problems is that they do not provide protection against the above mentioned Sybil attacks. Just as an example, [11] provides a mechanism to route DHT lookups beyond malicious nodes and show that if 20% of the nodes in the network are malicious, the lookup success rate is at 99.0%. These results are by far not sufficient for critical applications such as telephony. First, an attacker with sufficient resources could easily emulate more than 20% of the nodes in the network. Second, 99.0% lookup success rate is not sufficient since one aims at "5 nines" availability for telephony, i.e, an availability of 99.999%.

P2P systems deployed in practice today are actually rarely pure P2P networks. They generally do require some servers, even though most functionality can be performed among the peers. Even trackerless torrent functions with a server for the initial bootstrapping. However, in contrast to previous work on hybrid (or supervised) P2P networks, in this thesis a supervisor carries out a critical security function. It is considered as a *trusted third party*. The intention is to provide the same security guarantees as with server-based architectures. This is a requirement for critical applications such as telephony. On the other hand, one benefits from the resilience aspects of P2P networks and enhances the resilience of critical applications. Such a supervised P2P network can be considered as a solution between server-based and pure P2P-based signaling solutions leading to the combination of the advantages of both architectures.

1.1 Outline and Scientific Contributions

To guide the reader through this thesis, we provide a short summary of each part and highlight the contributions. Figure 1.1 provides an overview.

Part I of this thesis includes this introduction chapter followed by two background chapters. The first background chapter (Chapter 2) is

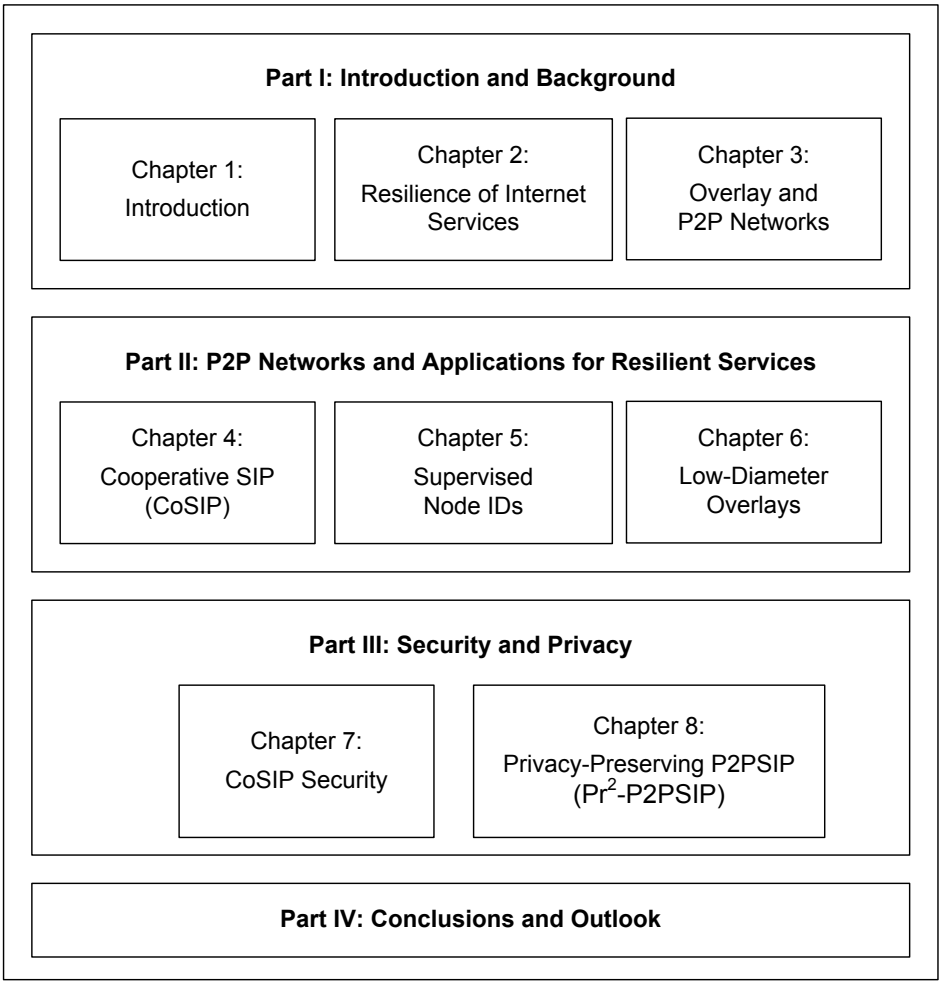


Fig. 1.1: Outline of this thesis.

about the state of the art of the resilience of Internet services . We shed light on the notion of *'resilience'* and related terms such as 'reliability', 'availability', 'dependability' and 'survivability', which are used throughout this thesis. We provide an overview of challenges in the Internet as well as resilience mechanisms. The second background chapter is about overlay and P2P networks. It introduces the appropriate terminology and describes P2P networks from algorithmic as well as operational perspectives.

Part II of this thesis provides contributions in the area of P2P networks, notably supervised P2P networks, and their applications for building resilient services.

In Chapter 4, we investigate the application of supervised P2P networks in the context of the Session Initiation Protocol (SIP). We present *Cooperative SIP (CoSIP)*, a hybrid architecture for SIP signaling which fills the gap between *i)* server-based telecommunication infrastructures (which are vulnerable to network and service faults, Denial-of-Service (DoS) attacks, etc.) and *ii)* pure P2P-based telecommunication networks (which are vulnerable to Sybil attacks, eclipse attacks, etc.).

Chapters 5 and 6 focus on the design flexibilities of P2P networks. In Chapter 5, we analyze how the P2P network can benefit from supervised node ID assignment. Given n peers in the network, a random node ID distribution used in pure P2P DHTs results into some peers being responsible for a higher load than the average load by a factor of $O(log\, n)$. A supervised P2P network approach like CoSIP guarantees a *near-optimal node ID distribution* where each node is responsible for a load in the order of the average load reducing the expected maximum load from a logarithmic to a constant factor.

In Chapter 6, we explore a further design flexibility in structured overlay networks. We present an algorithm for structured overlays where lookups can be performed within $O(1)$. More precisely, given random node IDs, routing can be achieved within two hops with a probability $(1-\epsilon)$ for an arbitrarily small ϵ. The routing table size is within $O(\sqrt{n})$. Then, the approach with supervised node IDs from Chapter 5 is applied here as well to increase the probability that routing within two hops succeeds to one.

Part III addresses the security issues which are raised by adding a P2P network in the CoSIP signaling procedure. Notably, we discuss whether the P2P network introduces new attack vectors and whether attackers are able to invalidate the resilience benefits of the P2P network. For example, storing an object at k peers provides redundancy, but becomes useless if all k peers are controlled by a single malicious node. We evaluate to what extent the supervised approach can solve the security issues inherent in P2P networks and discuss additional security mechanisms, e.g., resilient routing, to achieve the same security level as in server-based signaling solutions.

Finally, in CoSIP, storing user contact data in a P2P network raises also severe privacy issues. Thus, we learn from concepts and experiences from anonymization networks such as Tor [35] and I2P [39] to build *Privacy-Preserving P2PSIP* (*Pr*^{2}*-P2PSIP*) networks. The solution is valid for CoSIP as well as P2PSIP as being standardized in the IETF [98]. We provide a threat analysis as well as an analysis of the performance of Pr^2-P2PSIP Particularly we analyze cryptographic

overhead, signaling latency and reliability costs.

Table 1.1 summarizes the contributions of this thesis.

Tab. 1.1: Overview of contributions of this thesis.

Chapter	Previous Work	Contributions	Evaluation Methodology
Part II			
3	• Server-based SIP with lack of reliability. • P2PSIP with lack of security.	Cooperative SIP (CoSIP) combining advantages of both server and P2P-based architectures.	• Reliability Analysis based reliability theory and Skype traces. • Implementation. • Functional and performance evaluation on PlanetLab.
4	DHTs with random node IDs. Skewed load.	• Supervised node IDs. • Provably improved load balancing.	• Mathematical model. • Simulations.
5	DHTs with routing in $O(log\ n)$	DHTs with routing in $O(1)$	• Mathematical model. • Simulations.
Part III			
6	P2P networks with security issues.	Secure P2P networks with • Verifiable IDs and • Resilient routing.	• Security threat analysis. • Implementation of a selected security mechanism: SIP-based certificate enrollement.
7	CoSIP and P2PSIP networks with privacy issues: • Location privacy and • Social interaction privacy.	Pr^2-P2PSIP: A protocol which addresses privacy issues in CoSIP and P2PSIP.	• Threat analysis. • Analysis of cryptographic overhead, signaling latency and reliability costs.

2. RESILIENCE OF INTERNET SERVICES

Before we explore the potential of (supervised) P2P networks for building resilient network services, we first provide an overview of the state-of-the-art in building resilience Internet resiliences. In this chapter, we first shed light on the notion of *'resilience'* and related terms which are used throughout this thesis, for example, *reliability* and *availability*. Then, we provide an overview of different challenges in the Internet and appropriate resilience mechanisms. Given that the field of "resilient Internet services" is an emerging discipline with a large amount of effort, we do not claim to provide a complete list of appropriate resilience mechanisms. However, this chapter will help to position the resilience approach proposed in this thesis and compare it with other approaches. Furthermore, since this thesis is about *application layer* signaling, a special focus in this chapter is put on the resilience mechanisms at the application layer. Other mechanisms in underlying layers, e.g., IEEE link aggregation, IP Fast ReRoute (IPFRR), etc. are summarized very briefly.

2.1 Terminology

The term *resilience* in computer systems and networks is often used in relationship with other terms, e.g., *fault tolerance*, *dependability*, *availability*, *reliability*. These terms (including resilience) may sound synonymous or similar at a first glance. Nevertheless, there has been quite some effort by engineers and scientists to provide an appropriate taxonomy and clarify the differences. The IFIP working group (WG) "Dependable Computing and Fault Tolerance"[1]has been working on fault tolerance and dependability since 1980. One of the first results of this working group was a taxonomy document which has been updated iteratively [8, 73, 75]. Starting from the version of 1992 [75], the authors added security related terms and discussed the relationship between security and dependability.

A significant effort dedicated to definitions and taxonomies in the context of *network resilience* is lead by the Resilinets initiative[2]. Some of the results have been recently published by Sterbenz et al. [141].

We summarize the definitions of the relevant terms used in the literature

[1] http://www.dependability.org/wg10.4/
[2] http://wiki.ittc.ku.edu/resilinets/

in the context of resilience, notably network resilience. The effort by the IFIP WG and Sterbenz et al. will guide our terminology discussion.

2.1.1 The 'fault → error → failure' chain

Some of the basic terms that have been agreed on early in the disciplines of fault tolerance and dependability are *faults*, *errors* and *failures*:

- *Correct service* is delivered when the service implements the system function [8].
- A *service failure*, or simply *failure*, is an event that occurs when the delivered service deviates from correct service [8]. A service is a sequence of the system's external states. A service failure means that at least one external state of the system deviates from the correct service state.
- The deviation of an external state of the system from the correct service state is called an *error*. Thus, an error is the part of the total state of the system that may lead to its subsequent service failure [8].
- The adjudged or hypothesized cause of an error is called a *fault*. A fault is *active* when it causes an error, otherwise it is *dormant* [8].

These definitions lead to the causal relationship: fault → error → failure.

2.1.2 Fault Tolerance

Fault tolerance means to avoid service failures in the presence of faults [8]. Fault tolerance is one of oldest disciplines in computer science. For example, in 1956 Moore and Shannon provided an approach to build a reliable switching system out of relatively unreliable mechanical relays [91]. Jalote [59] binds fault tolerance to *redundancy*: "A system is fault tolerant if it can mask the presence of faults in the system by using redundancy". However, redundancy is not necessarily replication of the same object (software or hardware). For example, Laprie et al. [8] point that fault tolerance can be achieved via separate designs or implementations, i.e., *design or implementation diversity*. Further, one can differentiate between *hardware diversity* and *software diversity*.

2.1.3 Resilience

The origin of the term *Resilience* can be traced back to the Latin verb *resilire*, which roughly means "jump back". In physics, resilience is a material's property of being able to recover to a normal state after a deformation resulting from external forces. Laprie [74] provides different examples how the term 'resilience' is used in the context of psychology, ecology, business

and industrial safety. For example, in ecology, resilience has been used to refer to "moving from a stability domain to another one under the influence of disturbances"; in industrial safety, it has been used to refer to "anticipating risk changes before damage occurance".

The common denominator in the different usages of the term 'resilience' is the ability to successfully accommodate *changes*. Then, Laprie [74] defines resilience in computing systems as:

> *i) the persistence of the avoidance of failures that are unacceptably frequent or severe, when facing changes.*

Sterbenz et al. [141], define resilience in computer networks as:

> *ii) the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation.*

By "various faults", Sterbenz et al. [141] indicate that the faults may vary by type, scope, and number.

The definitions of resilience in computer systems by *i)* Laprie and *ii)* resilience in computer networks by Sterbenz et al. are not contradictory[3]. Both definitions accomodate the dynamic aspect of resilience; Laprie by the term 'change', Sterbenz by 'various faults'[4] and 'challenges'.

2.1.4 Dependability

The *dependability* of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable [8]. The concept of dependability has been developed over the last decades and encompasses:

- *availability*: readiness for correct service
- *reliability*: continuity of correct service
- *safety*: absence of catastrophic consequences on the user(s) and the environment
- *integrity*: absence of improper system alterations
- *maintainability*: ability to be restored after failure.

[3] An email discussion with the authors Sterbenz et al. lead also to this conclusion.

[4] Under the condition that these faults become active. Faults do not cause any change to the system as long as they are dormant.

Availability vs. Reliability

Reliability in computer networks has been defined in [149] as the probability of a network element (e.g., a node or a link) to be fully operational (i.e., absence of failures) during a certain time interval. Availability is the instantaneous counterpart of reliability: it is the probability of a network element to be operational at one particular point of time [149]. Both definitions of reliability and availability are consistent with the generic (i.e., not restricted to computer networks) definitions which can be found in reliability theory [111], except that the term 'network element' is generalized, e.g., by 'unit' or 'item'.

The definitions of availability and reliability allow for mathematical formulations which deserve a closer look here. In fact, we will use these formulations in the course of this thesis to estimate the reliability and availability of P2P networks, e.g., for P2P based SIP signaling in Chapters 4 and 8.

Let T be the *time to failure* of a unit, i.e., the time elapsed between the point of time when the unit is put into operation until the point of time when the unit fails for the first time. T can be assumed to be continuously distributed with a density function $f(t)$ and distribution function:

$$F(t) = P[T \leq t] = \int_0^t f(u)\, du \tag{2.1}$$

The reliability $R(t)$ is the probability that the unit will be still operating at time t

$$R(t) = 1 - F(t) = P[T > t] \tag{2.2}$$

The availability $A(t)$ is

$$A(t) = P[\text{unit is operating at time } t] \tag{2.3}$$

Series and Parallel Structures

We also would like to introduce some formulas which will be used in the course of this thesis to model the reliability of P2P networks (Chapters 4 and 8). A structure of units is *series* if the operation of the structure depends on the operation of all units in this structure. A *parallel* structure is a structure which operation requires at least one of the units operating. Let a structure consisting of k units with independent failures[5] and equal reliability distributions $R_i(t) = R(t)$ for all units $i = 1, \ldots, k$. If the structure is series, the reliability of the structure is

$$R_\wedge(t) = R_1(t)R_2(t)\ldots R_k(t) = R^k(t) \tag{2.4}$$

If the structure is parallel, the reliability of the structure is

$$\begin{aligned} R_\vee(t) &= 1 - (1 - R_1(t))(1 - R_2(t))\ldots(1 - R_k(t)) \\ &= 1 - (1 - R(t))^k \end{aligned} \tag{2.5}$$

[5] which is a dominant assumption in reliability theory

By estimating the reliability of single peers in the network, we can estimate the reliability of the whole system using equations (2.4) and (2.5).

Weibull Distribution

One of the commonly used distributions in reliability theory is the *Weibull distribution*. It is named after the Swedish professor Waloddi Weibull (1887–1979). The reliability of a unit is said to be Weibull distributed with a shape parameter $\alpha > 0$ and a scale parameter $\lambda > 0$ if the reliability function is given by

$$R(t) = P[T > t] = e^{-(\frac{t}{\lambda})^{\alpha}} \tag{2.6}$$

The Weibull distribution is used in this thesis to model the lifetime of peers participating in a P2P network (Chapters 4 and 8).

2.1.5 Survivability

The term 'survivability' was introduced in [38] denoting the "capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents". This definition has been re-used by Laprie et al. in [8] and slightly modified by Sterbenz et al. [141] as being the "capability of a system to fulfill its mission, in a timely manner, in the presence of threats such as attacks or large-scale natural disasters". The ambiguity with these definitions is that there is no quantification of 'accidents' or 'disasters'. Sterbenz et al. [141] draw the line between 'fault tolerance' and 'survivability' such as fault tolerance is the ability to cope with few and uncorrelated faults while survivability is the ability to also cope with many or correlated faults.

In this thesis, we use the term survivability in the context of SIP networks. A SIP network is survivable if session establishment remains possible in case of the failure of the SIP infrastructure.

2.1.6 Security

Security is defined by Laprie et al. [8] based the CIA model (*confidentiality*, *integrity* and *availability*).

- *Confidentiality*: absence of unauthorized disclosure of information.
- *Integrity*: absence of improper system alterations.
- *Availability*: readiness for correct service

Note that other security attributes can be composed using the three security attributes above and based on a given system (or network) specification. In the following, we provide three examples:

1. *Accountability* is the *availability* and *integrity* of the information which entity performed an operation.

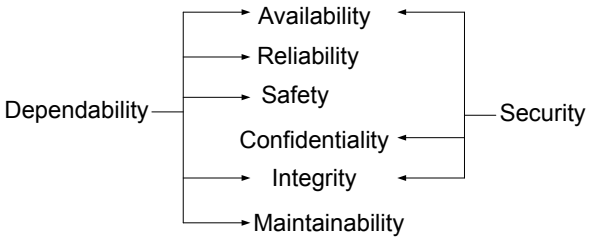


Fig. 2.1: Dependability and security attributes as defined by [8].

2. Let a system (or network) specification include access policies that regularize which subject is allowed to access which object. Then *access control* is

   - the *availability* to access certain objects for the authorized subjects
   - and the *integrity* of the system functionality according to the access policies.

3. *Non-repudiation* is the *availability* and *integrity* of the information which entity in the network has sent or received a message.

In fact, continuing with this reasoning, the three security attributes in the CIA model turn to be not orthogonal. Confidentiality is not an indispensable attribute in the definition of security and can rather be left to the system (or network) specification. For example, a web server can be fully secure while not providing confidentiality if the web content is supposed to be public. Thus,

- *Confidentiality* is the *integrity* of the system (or network) functions which regularize which information is to be disclosed to which entity.

2.1.7 Relationship between Security, Dependability and Resilience

Based on the CIA security model, we can see that dependability and security are closely related to each other. Figure 2.1 shows the common attributes between dependability and security. Dependability has originally been a discipline with a focus on faults which are introduced into the system without malicious objectives. But in many cases, regardless whether the root cause (fault or challenge) of a system failure is malicious or not, the same mechanisms may be applied first to recover the correct service, before a detailed look at the root cause can be taken. Over time, it has become clear that it would be a meaningful approach to consider dependability and security in joint efforts. Both disciplines deal with how the functionality of a system or

network can be impaired, and which *proactive* and *reactive* mechanisms can be deployed in order to achieve continuous system or network functionality.

The transition from dependability to resilience is the ability to accommodate *changes* as denoted by Laprie. In fact, in addition to the definition of resilience by Laprie quoted above, he provides a more concise, though equivalent definition:

- *Resilience is the persistence of dependability when facing changes.*

2.1.8 Conclusions

In this thesis, we adopt the definition of dependability by Laprie et al. [8] (Section 2.1.4). It has been widely used particularly by the Resilinets initiative [142]. We explained the difference between reliability and availability and how both metrics can be relevant for different network applications. Security may be reduced to the CIA model with the attributes integrity, confidentiality and availability. However, given that security addresses all kind of malicious actions that can be taken to break a system functionality or reduce its availability, it remains a tremendously wide topic.

While there has been no well established definition of resilience so far, we found that the definition by Laprie (persistence of dependability when facing changes) is concise and elegant. We do not see a contradiction to the definition of resilience by Sterbenz et al. [141]. Inline with these definitions, we consider resilience as a superset of disciplines which include security and dependability (and dependability itself includes reliability and availability).

2.2 Challenges

In the previous section, we mentioned that resilience is a superset of security and dependability. Thus, for example, a resilient SIP network would enable us to make phone calls even if the SIP servers or other related components, e.g., DNS, AAA, are encountering a DoS attack, the firewall in front of the SIP servers is misconfigured, or a cable in the backbone network is cut unintentionally due to construction work.

Apart from the straightforward classification of these challenges into malicious and non-malicious, we classify them in the following categories:

2.2.1 Topological Failures

Topological failures are failures of nodes or links in the network graph. They may lead to dis-connectivity between parts of the network. Furthermore, disconnections in some parts of the network may lead to an overload situation in other parts of the network. Topological failures can be, e.g., the failure of a submarine cable [69,70,103](physical layer) or a router (network layer). The impact of a router failure typically depends on whether the failure occurs

within an autonomous system. The reason is that the convergence time of intra-domain routing protocols, e.g., OSPF (i.e., the time until a new route is computed and can be used) is in the order in the 100ms [46] while BGP convergence time is in the order of several minutes [146].

Further, BGP misconfigurations can have a severe impact not only on the autonomous system where the misconfiguration occurs. A prominent example of BGP misconfigurations with a large impact is the one of YouTube and Pakistan Telecom [117]. Pakistan Telecom wanted to block YouTube. They announced a shortest path to the YouTube IP prefix, a so-called *black-hole route*. The BGP announcement propagated to other ASs worldwide. It made YouTube unavailable at a nearly global scale for about two hours.

2.2.2 Overload

Topological failures illustrated in the previous section are usually of a binary nature, i.e., a node or a link is either up or or failing. However, network equipment (including servers providing services over the network) have limited capacity, e.g., queue length, CPU power to process requests, etc. Thus, overload situations (also called congestions) are not rare and can render a segment of the network or a service over the network unusable.

One of the major challenges in countering overload situations is the difficulty to accurately differentiate between DoS attacks and legitimate but unusually high traffic, also called Flash Crowd. This differentiation is still extremely difficult despite continuous research in the area of anomaly detection [100].

2.2.3 Lack of Integrity

The vast majority of the Internet traffic today (signaling and data) is not protected from forgery. This is particularly the case for signaling protocols, e.g., DNS, BGP and email. Moreover, the lack of integrity of the system functionality goes beyond message integrity. Taking the YouTube example above, while the BGP update messages sent by Pakistan Telecom were not maliciously altered, the issue is that Pakistan Telecom is actually not eligible to sent updates for that BGP prefix.

2.2.4 Software Faults

Software faults can be classified into:

- Development faults introduced in the system during the development phase [8]. E.g., with buffer overflows in IP routers and
- Interaction faults introduced in the system after the development phase [8]. The YouTube example above is an example of interaction faults.

2.2.5 Correlated and Cascaded Failures

Different failures of parts or components in the network have often the same root cause. For example, a natural disaster may lead to power failure which leads itself to the failure of some routers or some servers in the network. Moreover, the natural disaster may lead to the cut of some fibre cables.

Furthermore, all kind of challenges above may trigger further challenges. For example, the failure of a server can easily lead to an overload of other servers providing the same service. The failure of DNS may lead to the unavailability of web servers although the web servers may be actually running and properly connected to the network.

2.3 Mechanisms

Since challenges may vary broadly from topology level link failures to application level message forgery (e.g., SIP), resilience mechanisms should be applied systematically at the different layers. Since this thesis is about *application layer* signaling, a special focus is put on the resilience mechanisms at the application layer here. Other mechanisms in underlying layers, e.g., redundant links, IP Fast ReRoute (IPFRR), etc. are summarized very briefly. Moreover, the deployment of integrity mechanisms, e.g., DNSSEC, BGP security, protection of SIP signaling, etc. should be considered where possible at all layers.

2.3.1 Protection at the Application Layer

Server Redundancy

It is straightforward that server redundancy is fundamental as a fault tolerance mechanism for the resilience of network services. In this section, we describe network mechanisms and protocols which can be deployed at different layers to support server redundancy. Some of these mechanisms can be deployed for servers connected within the same LAN. Others can be deployed for servers that are geographically distributed and thus allow for *geographic diversity*. Geographic diversity is particularly important in case of large scale natural disasters, e.g., earthquakes. Replicating a service in different distant locations, e.g., in different continents is the key solution.

IP Takeover

IP takeover is a resilience mechanism which can be realized with a simple topology for redundant servers. A backup server receives periodic keepalive messages from the master server as shown in Figure 2.2 at a sufficiently high frequency, e.g., every $10ms$. In case the backup server misses a keepalive message, it broadcasts an ARP message to the LAN via the Ethernet switch indicating the IP address of the master server is now reachable under the

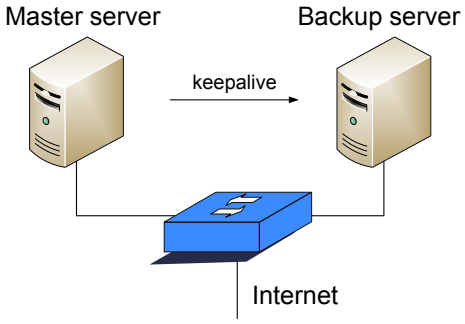


Fig. 2.2: Server redundancy with IP takeover.

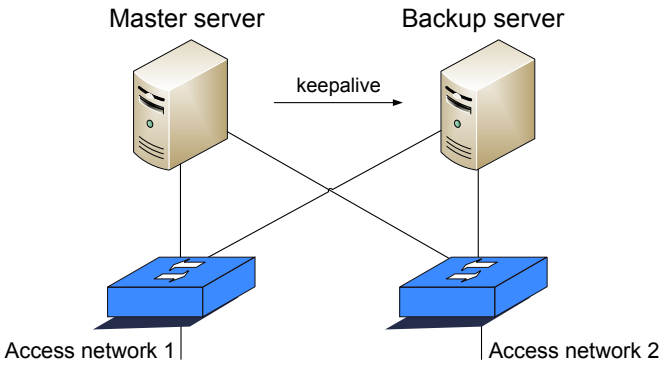


Fig. 2.3: Server redundancy with IP takeover and multiple switches.

MAC address of the backup server. From now on, all incoming traffic from the Internet to the master server is forwarded to the backup server. Obviously this means that any session state is lost unless the master and backup servers have additional state synchronization mechanisms.

One drawback of this architecture is that the Ethernet switch is a single point of failure. Figure 2.3 shows an extension to solve this problem. Here, both the master and the backup servers and connected to two Ethernet switches. The same procedure with ARP takes place when the backup servers misses a keep alive message from the master server, except that now traffic from both switches is forwarded to the backup server. The connection to the Internet with the two switches can be used, e.g., to connect to two different access networks, in which case the "server pool" becomes multihomed. Additionally, the Ethernet connections between the servers and the Ethernet switches can be enhanced with IEEE link aggregation (not shown in the Figure) which results in a topology where any components (server, Ethernet switch or cable) can be turn down, e.g., for maintenance

purposes, while the system keeps working. Under normal operation, link aggregation can be used to increase the bandwidth and the two servers can be also run in parallel for load balancing. Both solutions for IP takeover presented above require the redundant servers to run on the same Ethernet broadcast domain. Thus, they do not allow for geographic diversity per se, unless a layer-2 tunneling mechanism is deployed.

NAT Takeover

A similar takeover procedure as IP takeover can be implemented using Network Address Translators (NATs) which forward the traffic to a backup server in case the master server is not reachable. NAT takeover is not restricted to the LAN. Backup servers can be hosted anywhere in the Internet. Thus, NAT takeover allows for geographic diversity. A drawback of NAT takeover is that the NAT is a single point of failure. Thus, the resilience of a NAT-takeover-enabled server pool should be enhanced with additional mechanisms, e.g., IP takeover for multiple NATs.

Domain Name System (DNS)

Since hosts in the Internet are reached mostly using their host name and not the IP address, this indirection allows for multiple redundant servers with different IP addresses reachable behind the same host name. For example, consecutive DNS queries looking for a DNS *A* record for `www.youtube.com` provide different IP addresses in DNS responses. If a server becomes unreachable for any reason (i.e., either the server itself or its connectivity to the Internet is down), then its IP address is excluded from DNS responses. Moreover, in case of a flash crowd, additional servers can be added transparently and subsequent DNS responses can include the recently added servers. The process of allocating new servers can be performed along with the geographic client distribution along the network in order to optimize the latency for the clients. This is in fact the main idea behind Content Distribution Networks (CDNs) today. DNS allows for geographic diversity.

IP Anycast

IP anycast [4] has recently been used to provide access to replicated servers. Different servers in different locations can be addressed with the same IP address. Thus, IP anycast relies on IP routing to forward IP packets to one of these servers. For example, in BGP this can be achieved by announcing the same IP prefix from different origins. Figure 2.4 shows an example with a DNS *anycast cloud*. An anycast cloud is the set of servers available under the same IP address. In Figure 2.4, we can see two servers using the same IP address 192.5.5.241. DNS queries originating from different ASs may be routed towards different DNS servers.

Initial deployment of IP anycast for the DNS root servers started in 2002

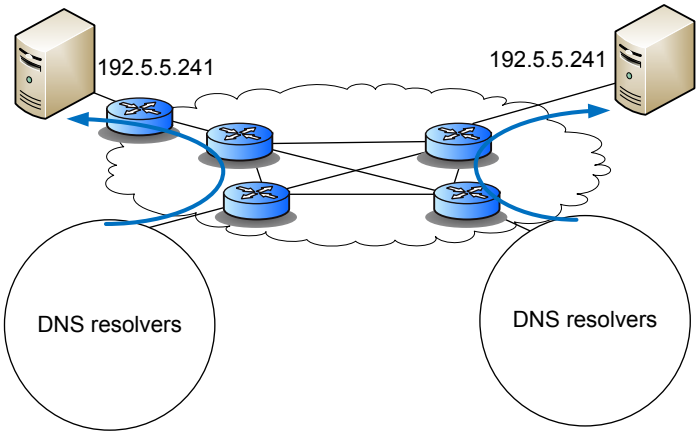


Fig. 2.4: DNS server redundancy with IP anycast.

with the F-root server. In the DNS case, IP anycast allows for easily adding new servers without the necessity to update the DNS client software. It renders more than 200 servers reachable worldwide[6] under 13 IP addresses. Each of the more than 200 servers may be internally compound of a load balancer and several physical servers, eventually deploying one the techniques for failover described above. IP anycast allows geographic diversity. Care must be taken if IP anycast is deployed with BGP since the MTTR depends on the routing convergences time, which may be up to several minutes in BGP. (See Section 2.2).

Virtualization

OS virtualization techniques, such as Xen, Kernel-based Virtual Machine (KVM) or VMWare enable an iterative relocation of the main memory of a virtual machine while it is still running, with an interruption of operation potentially below 100ms [68]. This can be exploited in several manners, e.g., to migrate a server to a network with more hardware and bandwidth capacity, or in the case of power failure (if the hardware can still run for a couple of minutes longer running on battery to enable the migration). One of the takeover techniques presented in Section 2.3.1 can be used to make services running within the virtual machine available as soon as the migration has been successfully completed.

Moreover, virtual machines can be duplicated and distributed along the network in order to cope with a higher load or to provide lower latency, such as the case in CDNs.

[6] 206 by 2010 July; with a trend to increase further.

2.3.2 Protection at the Lower Layers

Survivable Network Design

The network topology should be protected at different layers. At the phase of network planning, *Survivable Network Design (SND)* tools can optimize the network structure and basic operations. Potential topological challenges can be identified beforehand and serve, together with the network topology, as input to the tool, which then dimensions the network nodes and links [138].

Multihoming

Multihoming refers to a network setup where a host or a network is connected to the Internet via more than one connection. Multihoming can be applied in various contexts. For example, an IP host can be equipped with multiple network interfaces. Each of them may be connected to an other access network.

Multioming can be also applied at the transition point between different networks. For example, an enterprise network should be connected to different ISPs to ensure continuous Internet connectivity. An autonomous system is usually peering with multiple other autonomous systems to ensure connectivity to further autonomous systems in the globe.

Physical and Link Layer

Resilience mechanisms which can be applied at layer-1 and 2 are, e.g., protection switching, IEEE Link aggregation.

Protection Switching

Protection switching [52, 88] is a technique where backup paths are reserved proactively. In a failure case, a failover to the backup path can occur nearly without any delay.

IEEE Link aggregation

Link aggregation allows for bundling several physical Ethernet connections into a logical one. Devices supporting link aggregation, e.g., Ethernet switches or end hosts, connected by two parallel Ethernet links, can use the Link Aggregation Control Protocol (LACL) to double the available bandwidth between them. This bundling is transparent to the upper layers. Thus, a TCP connection can benefit of the double bandwidth. Apart from increasing bandwidth, IEEE 802.3d provides the possibility for redundant links inherently, and therefore, fault tolerance. One of the cables can be plugged out, e.g., for maintenance reasons, without interrupting the connection between the two devices.

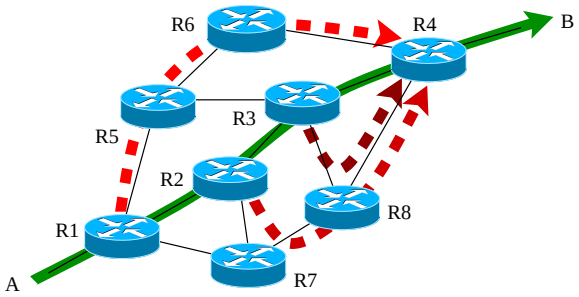


Fig. 2.5: Path diversity and IP fast reroute. Traffic from A to B crosses $R1$, $R2$, $R3$ and $R4$ under normal operation. $R1$ has a backup path to $R4$ via $R5$ and $R6$. $R2$ has a backup path to $R4$ via $R7$ and $R8$. $R3$ has a backup path to $R4$ via $R8$.

Network Layer

Resilience mechanisms should be applied at the network layer as well. For example, using *Traffic Engineering*, the routing can be optimized beforehand in order to cope with failure cases, such as IP routers or links between them do not become overloaded, neither under normal operation, not in case of challenges. Moreover, IP Fast Reroute (IPFRR) which has been standardized in the IETF, can be used to compute backup IP routes in advance such as routers can re-route traffic using the backup routes until the routing re-converges.

Multi Protocol Label Switching (MPLS)

IPFRR has been inspired by Fast Reroute in MPLS [58] which is switching technique used between the link layer and the network layer. Every MPLS switch has additional backup paths where the traffic can be rerouted in case of a hardware failure, e.g., the next MPLS hop or the link is down. The TTR can be kept below 50 ms [58].

In contrast to protection switching, where the source node needs to monitor the operation of a data path, notice the path failure and switch to the backup path, every MPLS switch with MPLS-FRR enabled can detect the failure of the next MPLS switch or the link and recover using a backup path. This can reduce the TTR by the order of a magnitude.

End-to-End Transport

TCP is the most widely used transport protocol in the Internet and one of the main resilience mechanisms currently deployed. It provides congestion control and mechanisms against packet loss and reordering. Moreover, TCP traffic is more likely to cross a misconfigured firewall and therefore offers better connectivity. It is worth it mentioning that while SIP is currently

often used on top of UDP, recent analysis advocate for the use of persistent TCP connections between SIP entities for improved resilience [96].

2.4 Reading Recommendations

For further details on the topic of this chapter, the reader is referred to the following documents.

- A comprehensive overview on the resilience-related terminology and various research activities on network resilience can be found in [141].
- An extensive taxonomy on dependability is provided in [8].
- The survey paper [6] provides a comparison of resilience related research activities.
- A good overview on current und future resilience mechanisms at the IP layer is provided in [52, 88, 108].
- Additional information on DNSSEC and MPLS can be found in [58].
- Finally, a nice-to-read general critique on the architecture of today's Internet is provided in [53], which addresses resilience aspects among other issues.

2.5 Conclusions

In this chapter, we shed light on the term 'resilience' and related terms such as dependability, availability, reliability and security. The terminology introduced in this chapter will help to understand the requirements on a resilient Internet application, e.g., SIP and will guide us to the appropriate evaluation methodology of the resilience concepts followed in this thesis.

We provided a classification of challenges for Internet applications. We provided an overview of different resilience mechanisms at different layers. Some of the described resilience mechanisms are already deployed today, e.g., multihoming, TCP congestion control and different takeover techniques for server redundancy. Others have different maturity levels, e.g., DNSSEC and IPFRR. Given the increasing requirements on the Internet and the frequency of partial failures, it can be concluded that there are still many open research questions regarding network resilience. Therefore, this topic will certainly remain interesting in the next years.

3. OVERLAY AND P2P NETWORKS

Different algorithms have been proposed and promoted in the research community for P2P networks and Distributed Hash Tables (DHTs) with different topologies, e.g., rings [144], trees [86, 123] and hypercubes [110], and different operational properties such as recursive vs. iterative routing, parallel lookups, underlay proximity, resource awareness etc. In this chapter, we provide a brief overview of the plethora of the design parameters for P2P networks with a special focus on *structured* P2P networks.

3.1 Terminology

Before we dive into the complexity of P2P networks and DHTs, we first revise the definitions of some basic terms.

3.1.1 P2P networks

The term 'peer' can be traced back to the Latin term 'par' which means 'equal'. P2P networks are generally *overlays* on top of existing networks, which allow for functionality not provided by the underlay, e.g., multicast or distributed lookup. In contrast to client/server systems with asymmetric roles, each peer in a P2P network implements the functions of both client and server.

3.1.2 Overlay Networks

"Overlay networks" is a term for network that run on top of existing infrastructure but provide certain additional functionality [1]. For example, Virtual Private Networks (VPNs) built using IPSec or TLS are overlay networks which enhance the underlay with encryption and integrity.

P2P networks are conventionally considered as overlay networks, since they perform routing at the application layer and can perform functions that are not provided by the underlying IP network, e.g., lookup and multicast. There are cases where a P2P network might not necessarily be considered as an overlay network. For example, a wireless ad hoc network. Nevertheless, this subtlety is not considered further in this thesis, since we focus on application layer signaling.

3.1.3 Unstructured Overlays

Unstructured overlays, e.g., Gnutella [2] organize nodes into a random graph topology and use floods or random walks to discover data stored by overlay nodes [26].

Given that unstructured overlays build random graphs, requests need to be flooded over the network. The number of messages grows exponentially by each hop. Limiting the number of flooded request may lead to not reaching some nodes in the overlay, although they are connected to the network. This means that if the P2P network is used for storage, it can not be guaranteed whether the lookup will succeed. Flooding lookup requests to all the participating nodes along all existing links has a time complexity of $O(n^2)$, where n is the number of peers in the network. This leads to inefficient network bandwidth usage.

3.1.4 Structured Overlays

Structured overlays impose constraints both on the topology of the overlay and on data placement to enable efficient discovery of data [26].

3.1.5 Distributed Hash Tables (DHTs)

We would like to underline a subtlety. Structured overlays and DHTs have been considered often as synonym. However, it is necessary to note that the term "structured overlay" does not necessarily imply that the overlay provides a storage functionality. In fact, peers in a structured overlay may cooperate for other purposes, e.g., multicast or anycast routing, publish-subscribe. This has been discussed, for example, in [29] where distributed storage is one of the purposes to build structured overlays. A DHT is thus an efficient distributed data structure implemented on top of a structured overlay network.

3.2 Algorithmic Aspects

In this section, we consider structured overlays in an abstract way. We basically consider them as graphs, and discuss their algorithmic aspects, which are mainly related to the topology how the overlay is built the routing algorithm. We illustrate the way how structured overlays and DHTs can be built based on examples of DHT algorithms described in the literature, notably Content Addressable Networks (CAN) [110], Chord [144], Pastry [123] and Kademlia [86].

3.2.1 ID Space

Each node in an overlay has an identifier from an identifier space $\mathcal{I}$.

Tab. 3.1: ID Space

CAN	d-dimensional Cartesian coordinate space $[0,1)^d$
Chord	$\{0,1\}^m$ interpreted as a integer in $\{0,1,\ldots,2^m-1\}$
Pastry	$\{0,1\}^m$ interpreted as a sequence of digits with base 2^b, e.g., $b=4$.
Kademlia	$\{0,1\}^m$ interpreted as a integer in $\{0,1,\ldots,2^m-1\}$

3.2.2 Distance Metric

Structured overlays requires a definition of a distance metric $d : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}$. Given that each node and each item is uniquely assigned to an identifier $x \in \mathcal{I}$, the distance metric d is used to determine the distance between two nodes, or the distance between a node and an item.

Tab. 3.2: Distance Metric

CAN	Euclidean
Chord	$d(x,y) = (y-x) \, mod \, 2^m$. Note that d in Chord is asymmetric, i.e., $d(x,y) = d(y,x)$ does generally not hold.
Pastry	Longest prefix match; Euclidean at the final search step
Kademlia	$d(x,y) = x \oplus y$ (bitwise XOR)

3.2.3 Consistent Hashing

Items which should be stored in a DHT are mapped to the identifier space. Typically an item *key* is computed first, e.g., by generating a cryptographic hash value from the item. Then the key is mapped to the identifier space. Nodes which are closest to the item key according to the distance metric are responsible for the item. A node storing an item is called *replica node* of that item. The set of nodes storing the same item is called the *replica set*.

The distribution of nodes along the identifier space and the data to be stored along the peers is critical for the performance of DHTs. *Consistent hashing* has been introduced in [65]. It aims for each peer receiving roughly the same number of items to be stored. It avoids the necessity of re-distributing the complete hash table each time a new machine joins the network. Any change in the network, e.g., node join and leave, requires only adjustments at a local scope. Consistent hashing is generally useful where multiple machines with different views of the network need to agree on common tasks without communication. Note however that consistent hashing suggests to use uniformly random identifier for nodes and items in a DHT.

This results into some unlucky nodes responsible for a higher load than the average load by a factor of $O(log\, n)$. In Chapter 5, we will provide and evaluate a supervised approach for node ID assignment where the expected maximum load is in the order of the average load, reducing the expected maximum load from a logarithmic to a constant factor. In the same time, it preserves the advantages of consistent hashing, i.e., any change in the network, e.g., nodes joins and leaves have a local impact.

3.2.4 Geometry

Algorithms for DHTs can be classified according to the geometry used for their presentation, e.g., rings, trees and hypercubes. The major design criteria are generally:

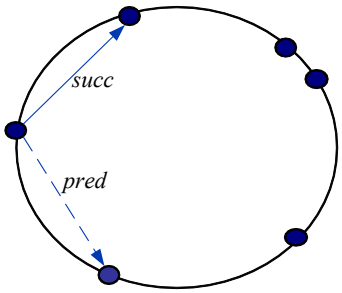
1. Lookup performance,
2. Routing table size,
3. Fault tolerance.

These design criteria are strongly interrelated. For example, in a full meshed overlay with a routing table size in $O(n)$ (design criterion 2.) any node can reach any other node with one hop (design criterion 1.). An additional advantage of this topology is that the graph is tolerant against link and node failures (design criterion 3.). Nevertheless, the problem with such overlays is that the routing table size is not scalable in many cases. Note that while the memory required to store a routing table may not be of a problem even with a couple of million peers. But the overlay maintenance (See section 3.3.2) may not be manageable.

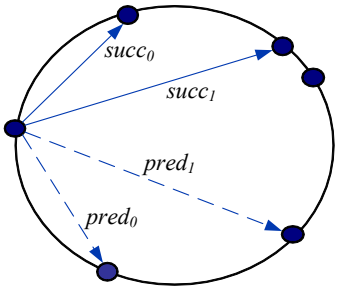
Another extreme example of a structured overlay is shown in Figure 3.1(a). Every node is connected to its successor and predecessor on a ring. Despite of being simple, a further advantage of this topology is that every node requires only to maintain a link to two other nodes (successor and predecessor) (design criterion 2.). However, the drawbacks are that:

- Lookups take $O(n)$ (design criterion 1.),
- The topology is not sufficiently fault-tolerant (design criterion 3.). The failure of only two nodes leads to a partitioned network.

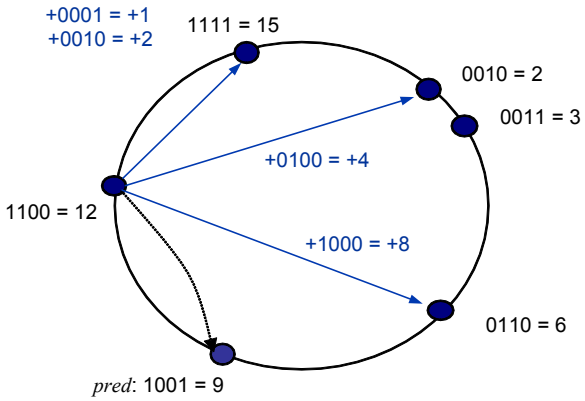
The discussion above shows roughly what must be taken care of when designing or choosing an appropriate topology for a structured overlay. We have observed that the two overlay design criterion 1. lookup performance and 2. routing table size require a tradeoff. In Chapter 6, we provide an overlay algorithm where routing can be achieved within $O(1)$, more precisely in 2-3 hops. The routing table size is $(2\sqrt{n})$.



(a) Simple ring: single *succ* and single *pred* for each node;



(b) Ring with multiple *succ* and *pred* for each node;



(c) Chord ring.

Fig. 3.1: Different ring-based overlay topologies.

The third criterion, fault tolerance, is a additional criteria which must be taken care of. Different metrics have been introduced in graph theory for evaluating graph fault tolerance, e.g., k-node connectivity, k-link connectivity and bisection width [47].

3.2.4.1 Rings

The ring structure may be the simplest DHT geometry. It is followed, e.g., by Chord [144]. Figure 3.1(a) shows a pure ring where each node in connected to its successor and predecessor on the ring. The drawbacks of this simple structure have been discussed above. In order to address the issue of fault tolerance, one may use multiple successors and predecessors for each node as shown in Figure 3.1(b). For example, each node connects to $l = 6$ successors and $l = 6$ predecessors (only two successors and predecessors are shown in

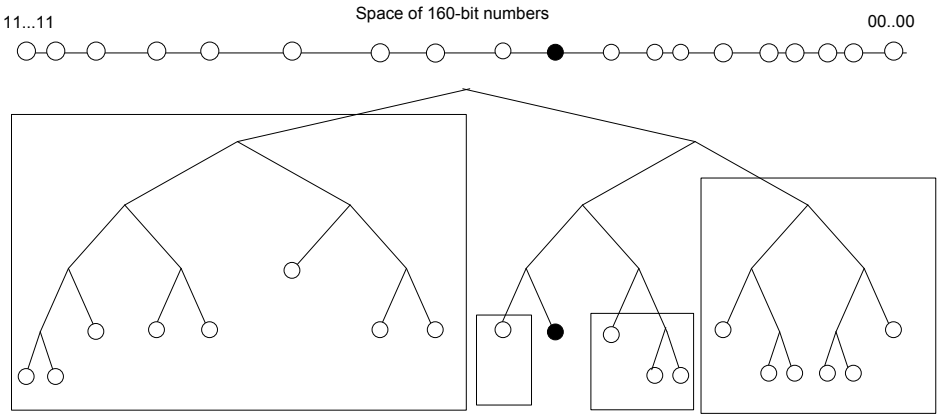


Fig. 3.2: Kademlia binary tree

Figure 3.1(b)). However, the lookup latency remains in $O(n)$.

In order to reduce the lookup latency, one may introduce additional links as *short cuts*. For example, in Chord a node with ID x_0 connects to the successors of the IDs $x_0 + 2^i$ for $i \in \{0, 1, \ldots, m-1\}$ (Figure 3.1(c)), m being the length of the binary presentation of IDs. Additionally, a node connects to its predecessor for maintenance reasons.

In [144], it has been proven that Chord lookup is within $O(log\, n)$ with high probability (w.h.p); and the routing table size is within $O(log\, n)$ w.h.p.

3.2.4.2 Trees

From a graph theoretic definition perspective, *trees* are "connected graphs with no cycles". A tree is a commonly used data structure, notably binary trees. Tress are well known to have a logarithmic average path length from the root to any other node. However, trees have some drawbacks if they are used in a distributed manner. First, the root of the tree (and nodes that are closer to the root) are a bottleneck in the routing (potential bottlenecks is in fact a 4^{th} design criterion which must be considered in DHTs). Second, the fault tolerance of trees is poor. It is sufficient to remove one node in the tree to partition the graph into multiple components.

Nevertheless, the structure of trees is borrowed in several DHTs. Links do not follow the strict graph theoretic definition of trees. Instead they cross-connect different sub-trees and provide short cuts between them. We consider the Kademlia DHT algorithm as an example (Figure 3.2). Nodes are mapped into the identifier space at the leafs of the tree.

Each subtree in the Kademlia tree can be uniquely identified by a prefix $\{0,1\}^l$ where $l = 0, 1, \ldots, m$. The routing table of a node in a Kademlia P2P network with ID x contains peers from the subtrees which differ with x in at least l bits, $l = 1, 2, \ldots, m-1$. In each of these subtrees, the node

keeps track of up of k contact peers. For example, the routing table of the black node in Figure 3.2 contains up to k contacts from each of the subtrees marked with a rectangle. The routing table in Kademlia is therefore called "k-buckets".

3.2.5 Routing

The main benefit of structured overlays is that the structure is used to efficiently route messages, e.g., DHT lookup messages. Routing from node a to an identifier x is generally *greedy* where node a chooses the *closest* node b to x in a's routing table according to the overlay distance metric. In case of routing failures, some DHT algorithms include a *backtracking* step. This means if routing is considered to fail at a certain routing step, the 2^{nd} closest node to x is chosen from a's routing table (or more generally the β closest nodes are chosen instead of the α closest nodes; where $\alpha < \beta$) and the routing procedure is re-initiated. This makes the routing procedure more resilient. For example, the Kademlia routing algorithm includes such backtracking.

3.3 Operational Aspects

In the previous section, we considered algorithmic aspects of structured overlays. We considered them mainly from a graph theoretic perspective. However, since peers join and leave the network, the design of a structured overlay needs to accommodate for the so called *churn*. Furthermore, the routing performance can be improved in terms of latency and success rate based on several optimizations discussed below. Additional aspects are connectivity across middleboxes and heterogeneous node capacities which have often lead to the differentiation between super peers and other nodes in practice. In the following subsections, we provide a brief overview of these issues which need to be considered when designing and building P2P networks.

3.3.1 Bootstrapping

Bootstrapping is the process of joining the P2P network. A peer joining the network needs to know the contact (i.e., IP and port) of at least another peer to join the network. At the very first time, e.g., when the peer software has just been installed and started for the first time, bootstrap requires an out-of-band mechanism to get informed about one or more other peers. For example, using a central server reachable via DNS. The server can, e.g., provide a list of currently online peers. Another method is to provide a list of peers with fixed IP addresses upon installation.

When a peer goes online after the initial bootstrap, it may use a peer cache with a list of previously seen peers. Another bootstrapping method

(also out-of-band) is to use broadcast to discover other peers in the broadcast domain, e.g. LAN.

3.3.2 Maintenance

Since peers join and leave the network gracefully and ungracefully, peers need to maintain the overlay topology and the DHT content.

Overlay Topology Maintenance

Peers need to refresh their routing tables regularly to maintain the anticipated overlay topology. The refreshing interval may be adjusted depending on the stability of the peers in the network. The whole routing table may be refreshed periodically. Additionally, maintenance can be done at a finer granularity. For example, if a peer b does not respond to a request from peer a, a detects that b is offline and may search for other nodes to fill the hole in its routing table. Further, different overlay operations may result into new information about other peers which can be added to the routing table as a side effect. For example, in Kademlia, peers benefit from lookup information to enrich their routing tables.

DHT Content Maintenance

An item stored in a DHT must be stored redundantly on different peers called *replica nodes*. The number of required replica node depends on the stability of the peers and the target reliability. A reliability model of P2P networks is developed in this thesis (Chapter 4) and provides a concept for computing the number of required replicas nodes to achieve a certain availability or reliability.

A difference in replica maintenance strategies can been observed in the DHT algorithms in the literature. Peers in Chord and Pastry rely on the replica nodes to re-shuffle data items in case new nodes join or existing nodes leave the network to make sure that the k closest nodes to an item key store the item. In Kademlia, the publisher of a data item is responsible for ensuring that the appropriate peers store the data item. Thus, the replica nodes do not have to re-shuffle any data upon joins and leaves. The impact of the different replication strategies is analyzed from security perspective in Chapter 7.

3.3.3 Parallel Lookups

Starting from the greedy routing algorithm as a basis, the routing performance can be improved in different ways. For example, node a may send the routing request to α nodes closest to x in parallel: $b_1, b_2, \ldots, b_\alpha$. For

example the Kademlia algorithm uses parallel lookups with $\alpha = 3$. Parallel lookups have been suggested for Chord as well in Epichord [79]. In the original paper of Chord [144] the routing failure rate is up to 8%. Epichord reduced the severity of this problem.

Furthermore, parallel lookups allow for embedding different routing optimization. For example, node a can choose different nodes $b_1, b_2, \ldots, b_\alpha$ according their location in the overlay as discussed below in Section 3.3.6.

3.3.4 Recursive vs. Iterative Routing

Using *recursive overlay* routing, a message is forwarded from a source A to a destination D hop by hop, e.g., via nodes B and C as follows:

$$A \rightarrow B \rightarrow C \rightarrow D$$

The response is either sent directly to A or hop-by-hop backwards taking the same path:

$$D \rightarrow C \rightarrow B \rightarrow A$$

In the latter case, the routing is denoted by *symmetric recursive routing*. Using *iterative routing*, the first hop on the path B does not forward the lookup message to C. Instead, it responds to A providing the contact data (IP and port) of C. Then, A contacts C directly and so on. The impact of recursive vs. iterative routing on the routing resilience will be discussed in Chapter 7.

3.3.5 Middlebox Traversal

Given that peers participating in a P2P network may be often running behind a firewall or a NAT, this may result into a lack of connectivity from a peer a to a peer b[1] where a should actually maintain a link to b. For this purpose, a wide range of techniques for middlebox traversal can be used, notably Simple Traversal of UDP (STUN) [121] and Interactive Connectivity Establishment (ICE) [119].

3.3.6 Underlay Proximity

Given that the overlay topology can be completely uncorrelated to the underlay topology, a message routed from a node a to another node b located in the node a's proximity in the underlay, e.g., in the same LAN may have to travel oversee several times before it reaches b. Therefore, it is generally useful to consider underlay proximity in the routing procedure. Gummadi et al. [49] differentiate between two strategies to integrate proximity in the routing.

[1] Links in a structured overlay are not necessarily bidirectional

- Proximity Neighbor Selection (PNS): The neighbors in the routing table are chosen based on their proximity.
- Proximity Route Selection (PRS): Once the routing table is chosen, the choice of the next-hop at each routing step takes proximity into consideration.

3.3.7 Resource awareness

Peers participating in a P2P network have often different capacities in terms of CPU, memory, network bandwidth and connectivity. It is critical for the performance of a structured overlay that 'weak' peers are excluded from the main overlay tasks such as routing and storage. This may result into a hierarchical approach with super peers and other non super peers. An evaluation of appropriate ratios of between the number of super peers and the number of non super peers is provided in [155].

3.3.8 Security

Given that a service is provided by the peers, e.g., file sharing or VoIP session establishment, it depends on the trustworthiness of the peers whether they perform their tasks, notably storage and routing, correctly and cooperate to provide a resilient service. If a P2P network is open to the public, which is the case for many file sharing networks, e.g., KAD [140], then a number of attacks are possible. Particularly *Sybil* [37] attacks where an attacker emulates a large number of peers in the network. Or *chosen-location attacks* where an attacker chooses its ID in the overlay such as it located in a strategically good location in the overlay, e.g., in the vicinity of their targeted victim peer. Or *eclipse* [135] attacks where attackers hide content or nodes in the network and prevent them from being reachable (eventually by cleverly combining Sybil with a chosen-location attacks). These attacks can render the resilience benefits of P2P networks invalid. Thus, appropriate measures are required to cope with these security issues. An extensive discussion of security threats in P2P networks and appropriate countermeasures is provided in Chapter 7.

3.4 Conclusions

In this chapter, we provided an introduction to the design of P2P networks with a special focus on DHTs. We separated between algorithmic aspects where P2P networks are considered from a graph theoretic perspective and operational aspects where deployment considerations are taken into account. Further investigations, notably in terms of availability, reliability, security and privacy are provided in the course of this thesis.

Part II

P2P NETWORKS AND APPLICATIONS FOR RESILIENT INTERNET SERVICES

4. COOOPERATIVE SIP (COSIP)

In Chapter 1, we introduced the notion of supervised P2P networks. In this chapter, we investigate the application of supervised P2P networks in the context of SIP. We introduce Cooperative SIP (CoSIP) a supervised P2P network approach which addresses the two critical resilience aspects of VoIP signaling namely reliability and security. CoSIP is a hybrid architecture based on a P2P network cooperating with central servers. The P2P network consists of SIP endpoints that organize themselves in a DHT. Both the DHT and the server manage user registration and session establishment in parallel.

While the DHT provides additional service reliability and robustness against DoS attacks, the server provides improved security and performance for the overall architecture. Our new architecture uses both technologies in parallel to combine advantages from both concepts, leading to improved reliability, security and performance.

The rest of this chapter is organized as follows. Section 4.1 presents background knowledge on SIP, P2PSIP and challenges in VoIP networks. Section 4.2 presents the CoSIP concept and example application scenarios. Section 4.4 provides an evaluation of the CoSIP approach. This includes a quantitative reliability analysis as well as functional and performance tests on PlanetLab. Section 4.5 presents related work and Section 4.6 concludes this chapter. For a security threat analysis of CoSIP, please refer to Part III of this thesis. Details of our CoSIP implementation including high-level state machines can be found in Appendix A.

4.1 Background

4.1.1 Session Initiation Protocol (SIP)

SIP [120] is a protocol standardized by the IETF for setting up multimedia sessions, notably VoIP sessions. It can also be used for Instant Messaging (IM) [118]. In the last few years SIP has dominated over the ITU H.323 protocol. It has been integrated into the 3GPP IP Multimedia Subsystem (IMS) [3].

The most relevant components of a SIP network are User Agents (UA), proxies and registrars. A SIP UA may either generate requests or process requests from other UAs. In the former case, the UA is called User Agent

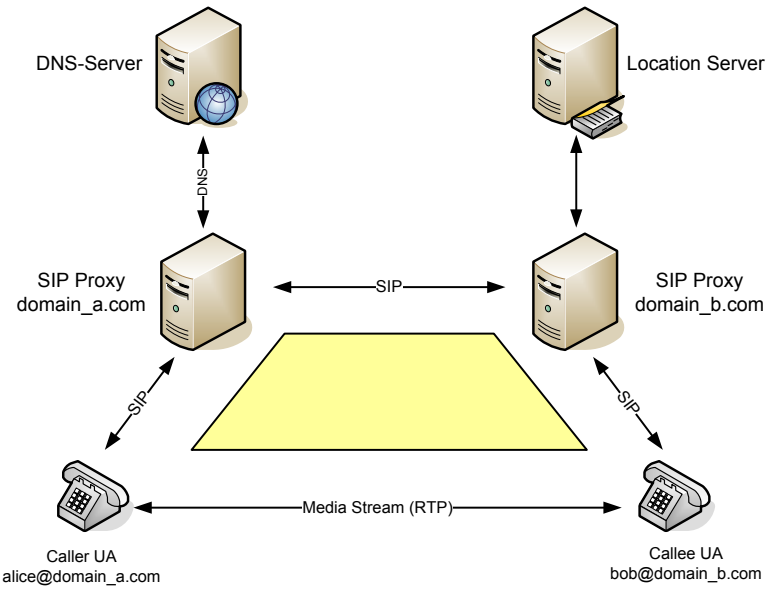


Fig. 4.1: SIP trapezoid.

Client (UAC). In the latter case, the UA is called a User Agent Server (UAS). A SIP registrar is a server that processes the registration of a UA to a certain location. The SIP registrar may use a location database and a AAA server to manage registrations and location information of its UAs. SIP proxies process and forward requests of SIP UAs, working together with SIP registrars in order to establish sessions between two UAs. Due to their close functionality, SIP registrars and proxies are often co-located together.

SIP uses of the Session Description Protocol (SDP) [54] to carry the session parameters, e.g., codecs, IP and port where to send the media streams. A SIP message consists of SIP headers and eventually a SDP body. Not all SIP messages carry a SDP body. For example, an `INVITE` request and the corresponding `200 OK` response carry the parameters for the session in the SDP body. A SIP `REGISTER` request and the corresponding `200 OK` response do not.

When the signaling for establishing the session is completed, UAs can start to send media data to each other. Media data can be exchanged directly between two UAs, typically using the Real-Time Protocol (RTP) [127]. In some cases, a relay node is used, e.g., due to NAT traversal problems, or if all the media data is routed via a single node for access control. Figure 4.1 summarizes the main functionality of "traditional" SIP. More details on SIP can be found, e.g., in [25, 62],

4.1.2 Challenges in VoIP/SIP Networks

Telephony is a critical service which needs to be protected in everyday life as well as particularly in case of large scale disasters. The transition from the PSTN/ISDN to VoIP and the use of the Internet as a common medium for telephony and data raises new resilience requirements on the Internet. Security threats that show up due to the use of the same medium for the audio/video data and the legacy Internet applications, e.g, DoS attacks, spam over IP telephony (SPIT), are among the major problems.

Apart from security, reliability is another major issue which deserves more attention with the wider deployment of VoIP. The complexity of the infrastructure for VoIP is one of the main reasons for this lack of reliability. The infrastructure includes SIP registrars, SIP proxies, AAA servers, DNS servers, DHCP servers, routers, firewalls and other network components, which require complex configuration. Even careful design leads frequently to inter-dependencies between these components. Network and service failures may propagate quickly. Moreover, reliability is affected by Denial-of-Service (DoS) attacks and flash crowds.

A number of incidents of VoIP service interruption have been reported in the press, e.g.,

- Subscribers of the German VoIP provider "1&1" experienced service failure on January 5th 2010 for up to four hours [82]. It is assumed that the root cause is a DoS attack on the DNS servers [17].
- Another service failure was experimented by "1&1" subscribers on November 11th 2009. According to the provider, the failure was due to network congestion and had to be remediated by manual traffic shaping [152].
- In August 2007, the P2P-based VoIP service Skype was unavailable for two days. Most of the functionality of Skype can occur in a P2P manner. However, the login process requires the Skype servers, which were unavailable. The businesses of many enterprises that used Skype on a daily basis were affected [137].
- A disruption in the infrastructure of the VoIP service at "United Internet" - a consortium of 4 large VoIP providers in Germany - occurred in July 11th 2006. It was not possible to make phone calls for two hours [13].

We mention also some other prominent examples of large scale failures of telephony services. These are not VoIP-based, but the root causes are software failures, e.g.,

- On April 21st, 2009, a failure during a Software update of the Home Location Register (HLR) of T-Mobile rendered the T-Mobile network

in Germany down for 4 hours [63]. Around 40 million subscribers were affected. During that time it was not possible to make phone calls, to send text messages or to use the Internet connections[1].

- The failure of a server at Deutsche Telekom on Monday October 29th 2007 [147] made phone calls to and from other providers unfeasible. Some calls were even routed incorrectly, i.e., phone calls were established to the wrong Callee.

These prominent examples indicate the clear need for better resilience, notably better reliability and security for telephony as a service.

4.1.3 P2P-based SIP (P2PSIP)

The basic idea of P2PSIP is as follows: Instead of using SIP servers, proxies and registrars which manage the location of users and moderate session establishment requests, P2PSIP uses a P2P network in which each node acts as a registrar.

For the illustration of the functionality of P2PSIP, we assume the usage of a DHT as a P2P network (in contrast to an unstructured overlay as the Kazaa network [76] and the Skype network [10]). The node identifier (node ID) in the DHT may be the hash value of its IP address, possibly combined with a port number. Another possibility is to use the hash value of the public key[2]. When Alice registers with her current location, the data `(alice_IP:alice_port)` is stored in the DHT under the key `H(alice@example.org)`, where `H` is a system-wide hash function. Then, Alice locates the nodes in the DHT who are responsible for the key `H(alice@example.org)`. Generally, nodes responsible for a key in a DHT are the nodes whose identifiers are the closest to the key according to the distance metric used in the DHT[3]. These nodes are called *replica nodes*. Alice asks the replica nodes to store her contact data. When Bob needs to establish a session with Alice, he sends a lookup message with the key `H(alice@example.org)`. The lookup message is propagated in the DHT until it reaches at least one of the replica nodes storing Alice's contact data. One or more replica nodes respond to the request. Then, Bob can initiate the SIP signaling directly with Alice without involving any servers.

Storage is based on a soft-state, i.e., Alice needs to refresh her contact data in the DHT periodically, e.g., once an hour. Otherwise, the replica

[1] unless people kept their mobile phone switched on and did not leave the same cell (3G or GSM cell). This is because in all other cases, re-authentication of the user based on the (U)SIM card is required, which involves the HLR, which was down

[2] In case of a supervised P2PSIP network as in CoSIP, the node ID is chosen by the supervisor. We recommend the supervised assignment of node IDs as proposed and evaluated in this thesis in Chapter 5.

[3] Different DHT algorithms use different distance metrics, e.g., Euclidean distance, Hamming distance, longest prefix match, etc.

nodes will consider the data as outdated and will delete it. In case Alice goes offline and some of the replica nodes are still storing Alice's contact data, Bob will be able to find Alice'contact data which were last stored in the DHT, but will not be able to reach Alice.

In March 2007, the IETF working group P2PSIP was chartered. The focus of the working group is to push SIP processing towards the endpoints and to keep the required infrastructure minimal, cf. [98]:
The Peer-to-Peer (P2P) Session Initiation Protocol working group (P2PSIP WG) is chartered to develop protocols and mechanisms for the use of the Session Initiation Protocol (SIP) in settings where the service of establishing and managing sessions is principally handled by a collection of intelligent endpoints, rather than centralized servers as in SIP as currently deployed.

David Bryan[4] published a document shortly afterwards with an overview of the motivations behind P2PSIP [21]. One of the main motivations is to decrease OPEX of VoIP providers. An other motivation is the easing of ad hoc communication. According to Schulzrinne [126], reliability is also a motivation given that P2PSIP should not depend on central components or DNS for session establishment.

4.2 CoSIP Overview

In this section, we present the CoSIP concept as well as example application scenarios.

4.2.1 Concept

Unlike the P2PSIP approach discussed in the IETF, we follow a supervised P2P network approach, combining the advantages of P2P and SIP client/server architectures in order to provide better reliability, security and performance. The SIP server cooperates with the SIP UAs to manage user registrations and session establishments. The current locations of UAs are stored at the server and are additionally propagated in the P2P network (Figure 4.2). Lookup requests are likewise resolved by the SIP server and by the P2P network concurrently (Figure 4.3).
As a choice for the P2P network, we use a DHT in order to achieve efficient routing and avoid flooding requests. Legacy SIP UAs that do not support CoSIP can be connected to the DHT via an adapter that we call "CoSIP proxy'" (see Section A). The same holds for weak devices with less CPU or bandwidth. They are represented in the P2P network by a CoSIP proxy running on a more powerful machine. In both cases, a CoSIP proxy processes user registrations as well as session establishments for the UAs behind it.

[4] The P2PSIP WG co-chair at the time of writing this thesis.

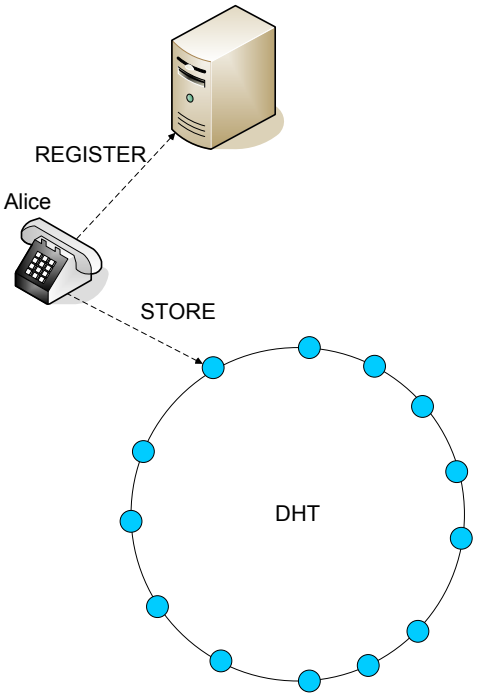


Fig. 4.2: Registration of a SIP UA with CoSIP.

Figure 4.2 and Figure 4.3 show the basic functionality of CoSIP. When Alice's UA registers to the SIP registrar, it also store her contact data in the DHT:

```
STORE(H(alice@example.org), alice_IP:alice_port)
```

This data will be propagated in the DHT and can be used afterwards to resolve the SIP URI of Alice to her current location. By the time another peer, let's say Bob, needs to initiate a session with Alice (Figure 4.3), Bob sends an `INVITE` message towards the SIP server. Additionally, Bob computes the hash value of Alice's URI and locate Alice's contact data in the DHT:

```
(alice_IP:alice_port) = GET(H(alice@example.org))
```

The `GET` request is propagated in the DHT until one of the replica nodes storing the data with the key `alice@example.org` responds.

A response from the DHT may take longer depending on the routing algorithm used for the DHT, the number of peers and the stability of the DHT. However, in case the server is unreachable due to a network or service failure, or the server is undergoing an attack or an overload situation, the response from the DHT can be very useful. When Bob receives the required information to contact Alice from the DHT, he sends an `INVITE` message

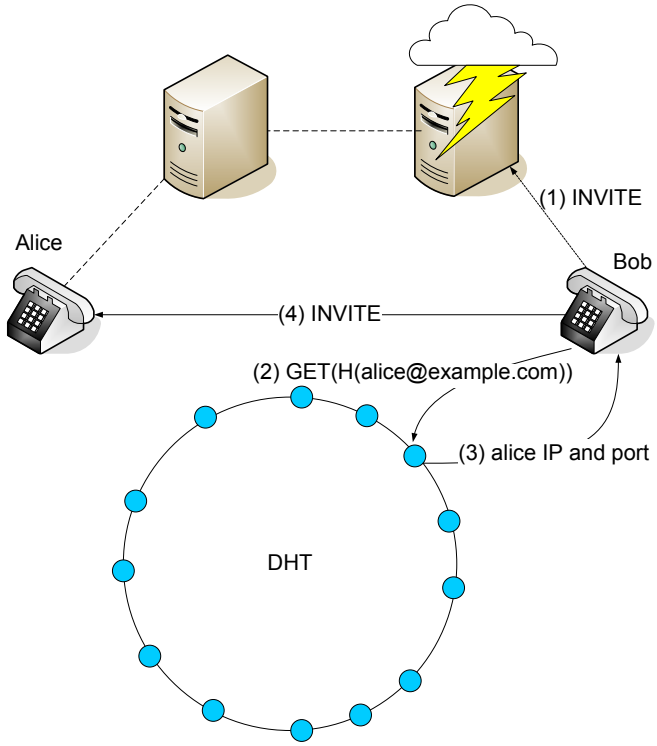


Fig. 4.3: CoSIP session establishment in case of a server failure.

directly towards Alice and the session establishment can be completed. In other words, in case the server is overloaded or unreachable, the DHT serves as backup. This provides a significant improvement to the reliability of the SIP service compared to traditional SIP. On the other side, CoSIP provides improved security compared to P2PSIP.

Concluding this section, centralized SIP infrastructures have a lack of reliability and are vulnerable to DoS attacks; P2PSIP networks are hard to secure and are vulnerable to a number of attacks such as Sybil attacks, eclipse attacks, partition attacks, or SPIT. Therefore, CoSIP is intended to fill the gap between these two solutions by combining them in order to benefit from their respective advantages.

4.2.2 Application Scenarios

An application scenario of CoSIP is sketched in Figure 4.4. A large-scale VoIP network, e.g., an enterprise or university SIP network. SIP UAs communicate with each other during normal operation and set up a DHT. Server downtimes due to failures or maintenance can be bridged by CoSIP.

Figure 4.5 shows another application scenario of CoSIP. Small Office and

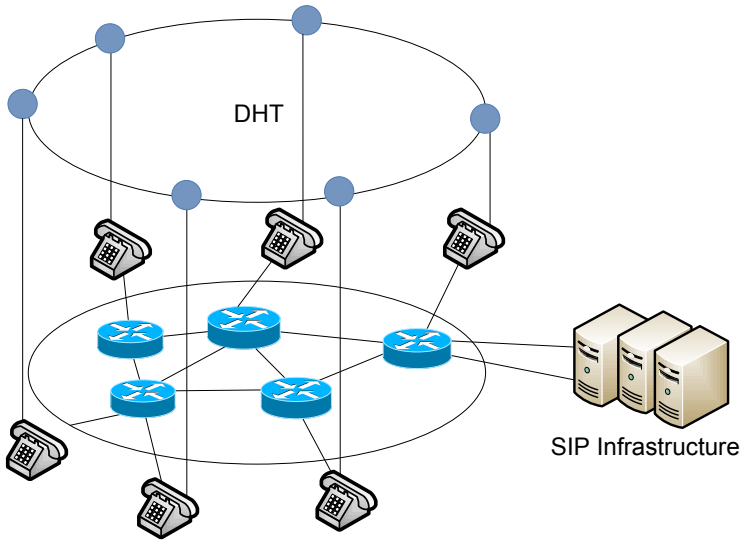


Fig. 4.4: CoSIP operation in an enterprise network.

Home Networks (SOHO) are connected to an Internet and VoIP provider via DSL routers. Each DSL router contains a CoSIP proxy which acts as an outbound proxy for end devices in its SOHO and implements the additional CoSIP functionality. The CoSIP proxies communicate with each other and organize themselves in a DHT. In the regular case, the SIP infrastructure of the VoIP provider is used to establish sessions between end devices in different SOHOs. In case the SIP infrastructure is temporarily unavailable, the DHT acts as backup and end devices can still establish phone calls.

In both scenarios described above, CoSIP provides a low-cost solution for significantly improving the reliability of the VoIP service. It is a proactive solution which does not require any challenge detection mechanisms or manual configuration when a failure at the infrastructure occurs.

4.3 Prototype Implementation

We implemented CoSIP as a local SIP proxy that processes the SIP signaling of one or more SIP UAs. Implementations of the SIP UA do not need to be aware of CoSIP. The SIP UA just needs to be configured with the CoSIP proxy as an outbound proxy. As a SIP server, we use the SIP Express Router (SER) [134]. The CoSIP proxy was implemented in Python. Figure 4.6 shows the implementation setup.

We successfully tested our CoSIP proxy implementation with SER as a server, and Kphone [139], Ekigacite [57] on Linux as well as XLite [56] and QuteCom (former WengoPhone) [106] on Windows. Figure 4.7 shows a screenshot of our CoSIP proxy implementation. The implementation details

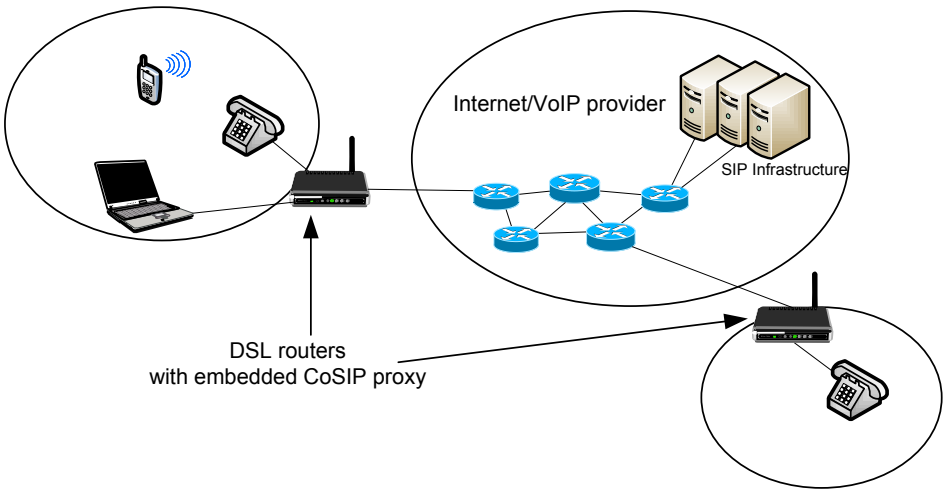


Fig. 4.5: CoSIP operation in an Internet/VoIP provider network.

are provided in Appendix A.

4.4 Evaluation

In this section, we provide an evaluation of CoSIP in terms of reliability, security and performance on PlanetLab. of this thesis.

4.4.1 Reliability Analysis

One of the main goals of CoSIP is to improve the reliability of the SIP signaling compared to server-based SIP signaling. In this section, we provide a quantitative analysis of the reliability of CoSIP based on reliability theory.

Reliability Theory

Reliability theory provides tools for estimating the reliability of a whole system by estimating the reliability of the single units/components of the system [111]. In Chapter 2, we introduced the mathematical definition of reliability and deduced appropriate formula for the reliability of a series structure (equation 2.4) and a parallel structure (2.5).

Modeling CoSIP with Reliability Theory

We model a CoSIP network as a system which consists of multiple units, which are the peers plus the server. The time to failure T of a peer is the time interval between the point of time when the peer goes online until the point of time when it leaves the network. In other words, T is the peer lifetime.

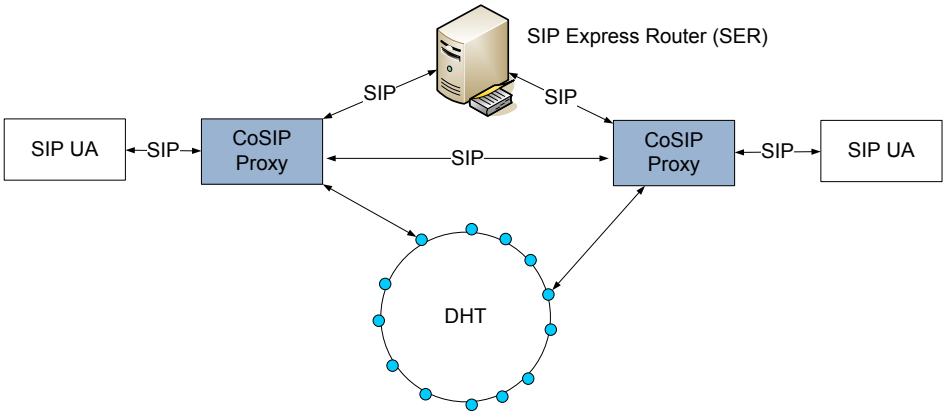


Fig. 4.6: Architecture of a SIP network with CoSIP proxies.

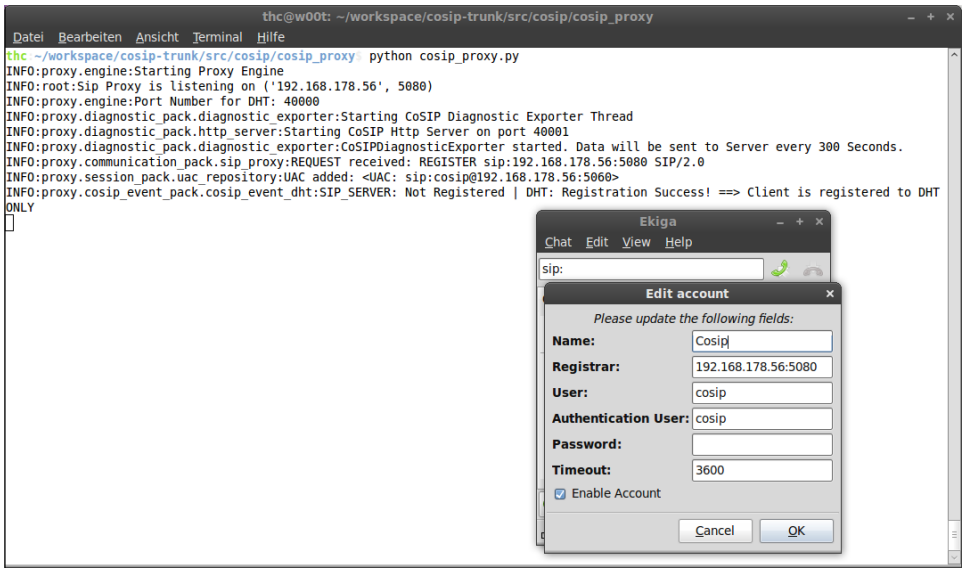


Fig. 4.7: Screenshot with a SIP UA (Ekiga) and a CoSIP proxy running in P2P mode in the background.

Modeling the reliability of CoSIP is a challenging task for different reasons:

- Modeling the reliability of the server: we are actually considering a complex system which consists eventually of multiple SIP servers, registrars, proxies, DNS servers, AAA servers, firewalls, etc. as a single unit. Modeling the reliability of such a complex system is not possible if the reliability of the single units within that system and the exact dependencies between them are unknown. Therefore, we will restrict our analysis to the case where the server is unreachable by the SIP endpoints for any reason, and compute the reliability of CoSIP in case of server failure.

- Modeling the reliability of the peers: There has been different studies of the peer lifetime in P2P networks, notably KAD [140] and Skype [48]. However, it is not straightforward that any of these studies is useful to model the peer lifetime in a CoSIP network. KAD is a file sharing network and not VoIP. Skype is a VoIP application. But it is running mainly on PCs/laptops. Thus, Skype shows a high number of online peers during working days and middays, while peers in a CoSIP network could be running, e.g., on some fixed hardphones which are permanently online, or on mobile smart phones, which may change their IP addresses frequently. Nevertheless, Skype is the most similar application to CoSIP where measurement data are available. The Skype traces published in [48] will help us to estimate the reliability of CoSIP. The reliability analysis of the KAD network in [140] will be also helpful for a comparison.

A Reliability Model based on Skype Traces

Guha et al. [48] collected traces by monitoring 4000 nodes participating in the Skype supernode network for one month beginning September 12, 2005. The traces are available under `http://www.cs.uiuc.edu/homes/pbg/availability/`. The authors in [48] plot the complementary CDF (CCDF) of the lifetime of the Skype supernodes in order to detect a power-law relationship. We use the same traces to detect whether the supernode lifetime can be modeled with a Weibull distribution instead. Figure 4.8 shows the CCDF of the supernode lifetime together with a Weibull fit and and a power-law fit.

It is clear from Figure 4.8 that the Weibull fit is better suited than a power-law fit. The scale λ and the shape α parameters of the Weibull distribution are approximately equal to 8.84 and 0.52 respectively (See equation 2.6 for the definition of the Weibull distribution).

From the definition of reliability (See equation 2.2), it follows straight-

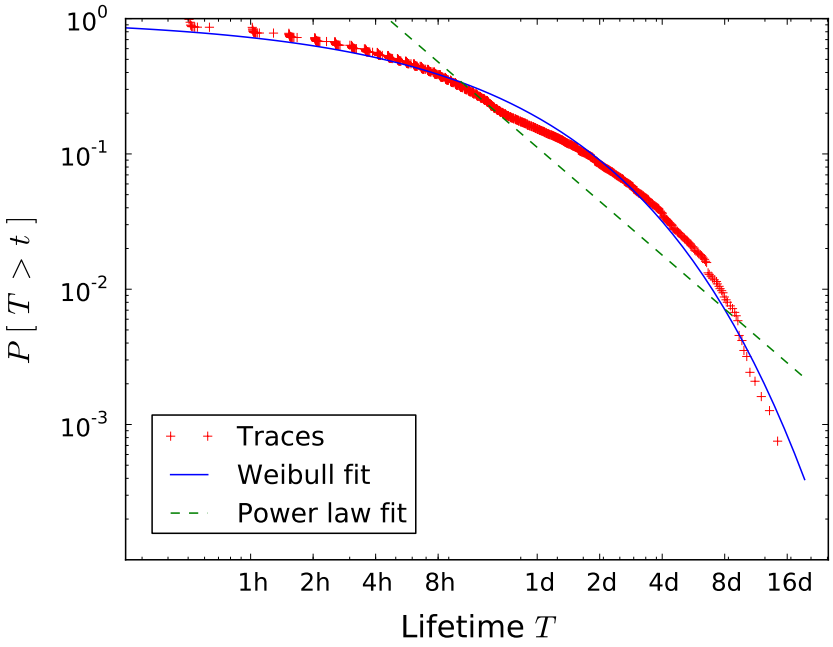


Fig. 4.8: Complementary CDF of the supernode lifetime distribution in Skype.

forward that the CCDF of the supernode lifetime is its reliability:

$$P[T > t] = R(t) \tag{4.1}$$

Thus, the reliability of a Skype supernode can be modeled based on the Weibull distribution

$$R_{Skype\_supernode}(t) = e^{-(\frac{t}{\lambda})^{\alpha}}; \quad \alpha = 0.52 \quad \lambda = 8.84 \tag{4.2}$$

The median peer life time is about 5.1 hours. Note that the median in this case is greater than the median value reported for the KAD network [140] which is about 3 hours ($181min$). The reason is that the Skype traces of Guha et al. include only the Skype *supernodes* which are more stable than the average peer. According to [140], the peer lifetime in KAD follows a Weibull distribution

$$R_{KAD\_peer}(t) = e^{-(\frac{t}{\lambda})^{\alpha}}; \quad \alpha = 0.545 \quad \lambda = 5.95 \tag{4.3}$$

Figure 4.9 shows the difference between the estimated reliability of KAD peers according to the model derived in [140] and the estimated reliability of Skype supernodes according to the Weibull fit that we derived above.

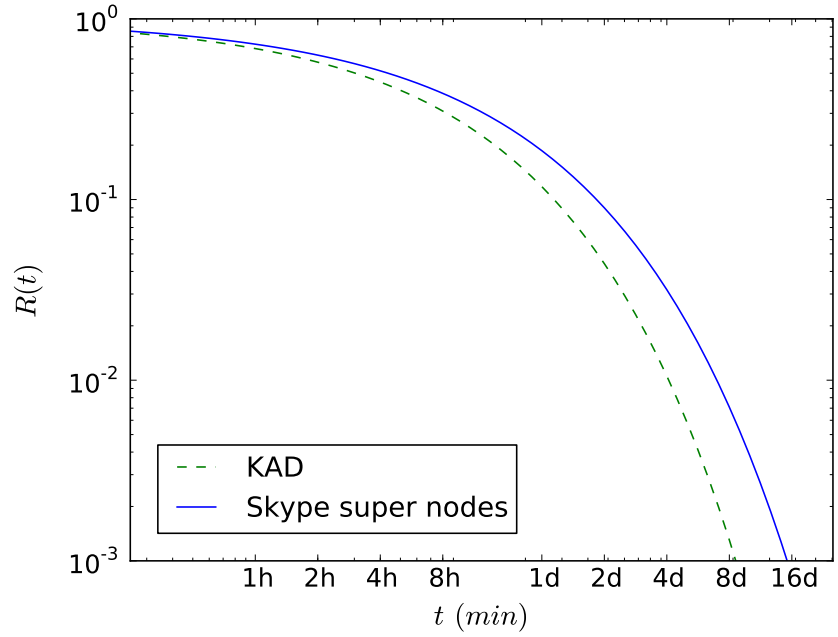


Fig. 4.9: Comparison of the reliability of Skype supernodes and KAD peers.

CoSIP Reliability Model

Let $R(t)$ be the reliability of an arbitrary CoSIP peer at a point of time t, i.e., the probability that an arbitrary peer is online until t. A UA Bob refreshes his contact data in the DHT periodically (with a `STORE` RPC) with a refreshing period e.g., $\tau = 1h$, in order to make sure he remains reachable in the CoSIP network with high probability. If Bob sends a `STORE` RPC to a replica node Carol at $t = k\tau, k \in \mathbb{N}$, and receives an acknowledgment message from Carol, Bob can deduce that Carol is online[5]. Thus, the probability that the peer Carol is online at that point of time $t = k\tau$ is equal to one. Then, this probability decreases over the time to the minimum value at the end of the refreshing period. An example of this behavior with a refreshing period $\tau = 1h$ is shown in Figure 4.10. Let μ be the minimum reliability of an arbitrary peer at the end of each refreshing period τ:

$$\mu = \liminf_{t \to (k+1)\tau} R(t) \quad k \in \mathbb{N} \tag{4.4}$$

μ can be estimated autonomously by Bob through measurements. It is the probability that if another UA (peer) is observed online at a point of time t, the UA will remain online until $(t + \tau)$. The value μ can be considered as a

[5] Under the assumption that the acknowledgment message is integrity-protected.

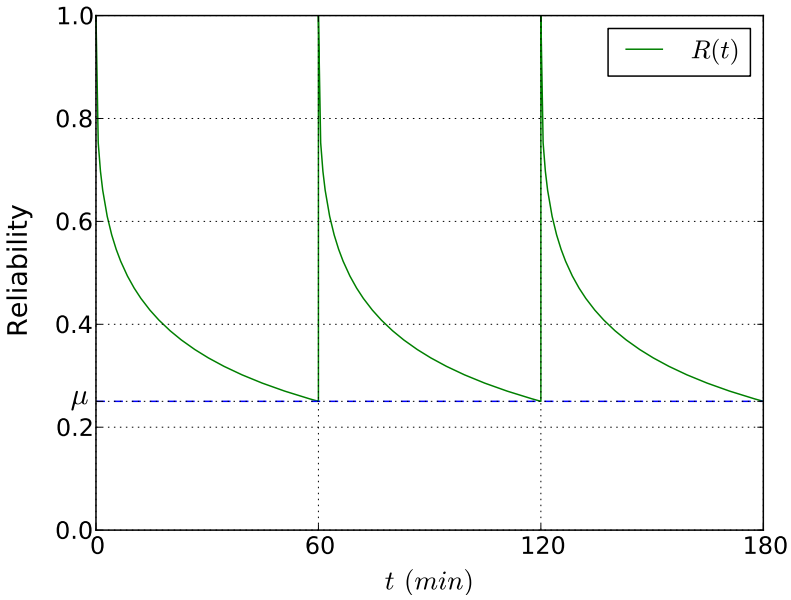


Fig. 4.10: Example reliability of a single peer p_i with periodic refresh. $\tau = 1h$.

metric for the churn in the network. If the measured μ is too low, then the UA may have to decrease the refreshing period τ, and thus increasing μ.

Furthermore, the following assumptions are required for our reliability analysis:

- We assume that all peers are cooperative, i.e., as long as a peer is online, it will perform storage requests from other peers.
- We assume that peers/UAs leave and join the network independently. A UA which leaves the network deletes all contact data of other peers.
- We assume a DHT model like in KAD [140] where peers which publish data are responsible for refreshing this data themselves, i.e., replica nodes do not re-publish data among each other, in particular when some of them leave the network, or new nodes close to the key of the data enter the network. As we discuss later in Chapter 7, this model is efficient to counter churn attacks.
- We assume that routing in the DHT always succeeds. In particular if Alice is looking for the contact data of Bob and there is at least one replica node p_i storing this data, then Alice will be able to reach p_i and find the contact data of Bob.

Figure 4.11 shows the resulting reliability model under these assumptions. A UA Alice calling Bob needs to reach either the server S or at least

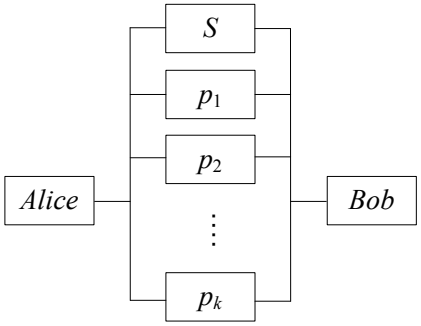


Fig. 4.11: CoSIP reliability model.

one of the storage peers p_i which have stored the contact data of Bob. Thus, assuming uncorrelated failures of the SIP server and the peers in the DHT, the reliability of CoSIP is:

$$R_{CoSIP}(t) = 1 - (1 - R_S(t)) \cdot (1 - R(t))^k \tag{4.5}$$

where $R_S(t)$ is the reliability of the server, $R(t)$ is the reliability of an arbitrary peer, and k is the number of replica nodes.

As mentioned above, R_S is difficult to estimate without knowing much about the service topology and the reliability of the single units providing the service. Thus, we restrict our reliability analysis to the case where the SIP server fails, which is the aggregated reliability of the k replica nodes storing Bob's contact data:

$$R_{replica}(t) = 1 - (1 - R(t))^k \tag{4.6}$$

To model the reliability of arbitrary peers $R(t)$, we use the parameters that we deduced from the Weibull fit of the Skype supernode lifetime above (Equation 4.2). Figure 4.12 shows the estimated reliability $R_{replica}(t)$ with different number of replica nodes k. Since the Skype model that we have is based on traces of supernodes only, we provide also the estimated reliability based on the KAD model (Equation 4.3) in Figure 4.13.

While $R_{replica}(t)$ is slightly higher for the Skype-based model (the difference can be observed, e.g., at the end of the refreshing periods, $1h$, $2h$ and $3h$) in both cases high reliability can be achieved with a quite small number of peers. For example, a reliability of "3 nines" can be achieved with $k = 6$ and a reliability of "5 nines" can be achieved with $k = 10$, in both cases with a refreshing period of $1h$.

Pitfalls

The reliability model in Figure 4.11 is based on some assumptions mentioned above. We now provide some comments on the validity these assumptions.

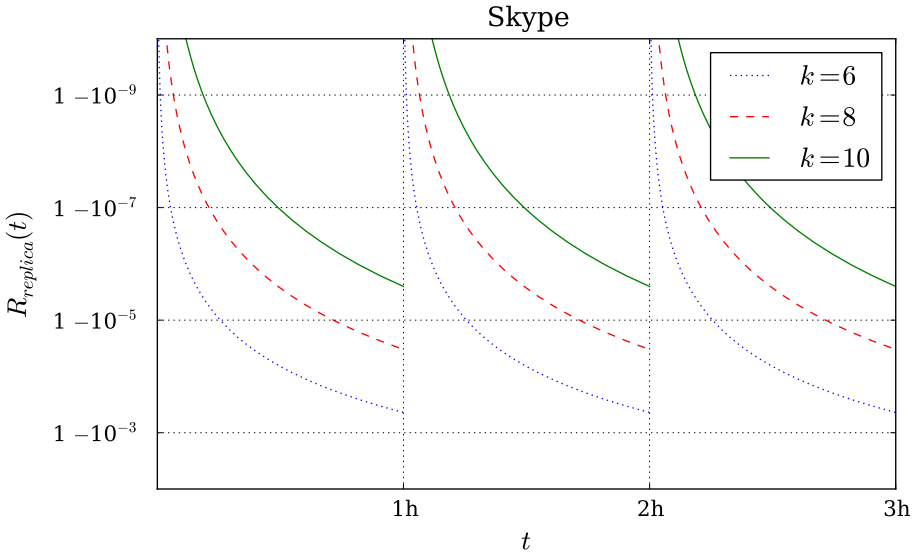


Fig. 4.12: Reliability of k storage peers in parallel using Skype supernodes as a model.

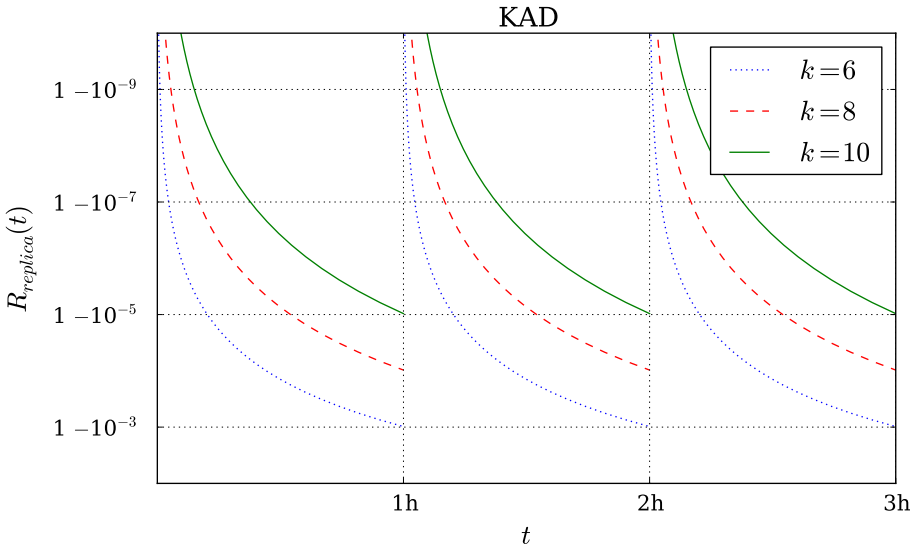


Fig. 4.13: Reliability of k storage peers in parallel using KAD peers as a model.

One assumption that we made for the reliability model is that routing always succeeds. CoSIP uses currently the Kademlia DHT algorithm. Using Kademlia, a peer looking for a data item in the network can probe parallel paths to the destination. The Kademlia routing algorithm has proven to be robust [64]. Even if a large number of peers leave the network within a short timeframe (up tp 90%), it has been shown that the Kademlia overlay can recover efficiently [16]. Moreover, the connectivity in the overlay can be enhanced if low-diameter overlays with degree in $O(\sqrt{n})$ instead of $O(log\, n)$ are used as described in Chapter 6. For these reasons, we left the routing reliability out of scope. Nevertheless, successful routing in the overlay requires adequate protection from Sybil and eclipse attacks. This will be the subject of Chapter 7.

The assumption that all peers are cooperative is not certain. Malicious peers may behave inconsistently, e.g., by responding positively to `STORE` requests but just discarding the data they are supposed to store, or ignoring subsequent `GET` request. Thus, the overlay must allow for sufficient redundancy in routing and storage to counter faulty behavior. Moreover, the overlay must allow for protection against Sybil attacks, which could render redundancy useless.

Finally, we assume uncorrelated failures of the replica nodes. This can notably not be the case if a node launches a successful Sybil attack. A host node may send a `STORE` request to k virtual nodes, thinking that it benefits from high reliability while a malicious node may be running $j \leq k$ nodes, thus invalidating the reliability assumptions of the honest node. Therefore, this is another reason why protection against Sybil attacks.

To summarize, the reliability and security of CoSIP are strongly interrelated. This emphasizes the relationship between reliability and security in the overall context of resilience. The security issues of CoSIP are discussed further in Chapter 7.

4.4.2 Security

P2P networks lack a central control and are thus prone to several kinds of attacks, for example, attacks on the overlay routing or DHT content. Being a supervised P2P network approach, CoSIP provides peers with verifiable identities

- at the application layer, i.e., SIP URIs embedded in an X.509 certificate,
- at the overlay layer, i.e., node ID embedded in an X.509 certificate.

An extensive evaluation of the impact of such a supervised approach with verifiable identites on the security of the P2P network is provided separately in Chapter 7.

4.4.3 Evaluation on PlanetLab

In this section, we report results from experiments with our CoSIP implementation. We performed extensive experiments on local testbeds as well as on PlanetLab to validate the functionality of CoSIP. PlanetLab is a research network that can be accessed by members of associated organizations for testing new services [104]. The intention behind PlanetLab was having an emulated "real life-like" network. One of the most remarkable properties that we noticed when experimenting with PlanetLab was the high variance in the round trip time (RTT) due to the different locations of the nodes and the different load on the respective machines. A limitation of PlanetLab is that we can not run a realistic VoIP network of an Internet/VoIP provider given the limited number of PlanetLab nodes.

Testbed Setup

The main motivation behind our tests was to show a survivability use-case of the SIP service, i.e., if CoSIP is used, SIP UAs are able to establish a session even if the server is unreachable. Additionally, we measured the time required to establish a session in two cases: first, under normal conditions with server-based signaling, and second, with an emulated server failure and DHT-based signaling. We compared the time for the session establishment in both cases. We consider the time required for the session establishment as the length of the time interval between sending an `INVITE` message by a UAC, and receiving a `180 Ringing` message from the UAS[6].

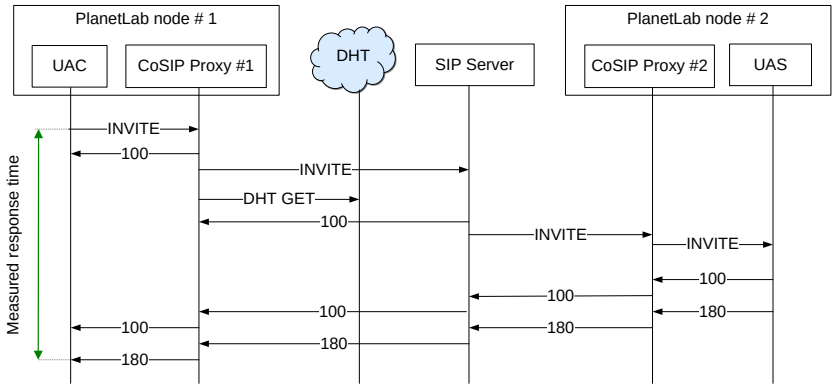
We use the open source SIP stack PJSIP [102] to generate SIP traffic. We use the PJSIP API to perform UA registration to the SIP server and initiate VoIP sessions randomly. We log timestamps when a message is sent or received at the CoSIP proxy. We run a network of 430 PlanetLab nodes with a SIP UA and a CoSIP proxy on each PlanetLab node. Given the relatively small size of the network we performed different experiments with different size of the Kademlia k-buckets, particularly $k = 2, 4, 8$. The SIP Server (SIP Express Router) and a Kademlia node for bootstrapping the DHT nodes were installed on a server running in our department at TU Munich.

Functional Tests

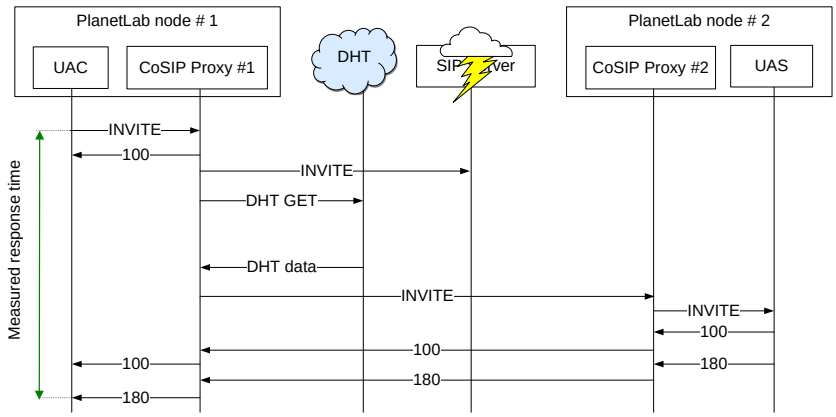
Figure 4.14 shows the message flow for session establishment with a running SIP server and when the server is unreachable respectively. In both cases, message flows were collected based on logs from the CoSIP proxies. The

[6] Waiting until a `200 OK` is received would not be appropriate as a measure, since it depends on how long the user needs to pick up the handset. Therefore, the `180 Ringing` message seems to be more appropriate as a sign that a session between the UAC and the UAS has successfully been established.

message flow in Figure 4.14(b) shows that the SIP network can survive and that SIP UAs are able to establish a VoIP session even when the server is down.



(a) SIP server up



(b) SIP server down

Fig. 4.14: Message flow for session establishment

Performance Tests

After a test run for 5 hours, 5,380 phone calls were emulated by the UAs on PlanetLab. All logs are exported by the CoSIP proxies periodically to a central server which collects the call records in a SQL database. Figure 4.15 shows the time required for session establishment with the server vs. DHT with different k-bucket size $k = 2, 4, 8$. Each boxplot shows the 10%, 25%,

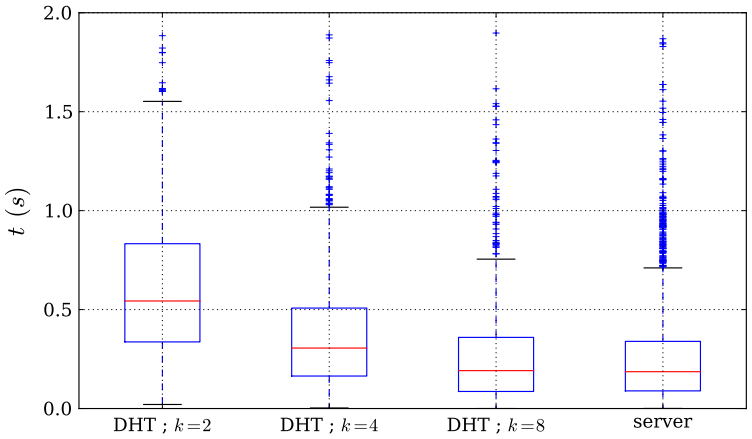


Fig. 4.15: Time required for session establishment with server vs. DHT with different Kademlia k-bucket size.

50%, 75% and 90% percentile. The median is approximately 0.543, 0.306 and 0.191 for the DHT with $k = 2, 4, 8$ respectively and 0.186 for the server The difference between the median values for $k = 8$ and for the server case is quite small (less than $1ms$) given that in the majority of the cases the DHT lookup can be performed within one hop.

4.5 Related Work

The goals of CoSIP are to improve *i)* reliability compared to server-based SIP and *ii)* security compared to P2PSIP. Security issues in P2P networks, notably P2P for VoIP signaling, are discussed in this thesis in Chapter 7. Thus, related work on security in P2P and P2PSIP networks is discussed separately in that Chapter. In this section, we focus on signaling approaches which are similar to CoSIP, notably those which use a P2P network for signaling.

P2PSIP

P2PSIP has been initially proposed by [22] and [132] and raised much interest and follow up work. In 2010, at IPTComm, the conference dedicated to IP telephony[7], there was a separate session for P2P telephony. RESource LOcation And Discovery (RELOAD) [61] is the IETF base protocol for P2PSIP. It is a generic P2P protocol. It allows for different overlay algorithms to be plugged in and different 'usages'. This means it is a P2P

[7] http://iptcomm.org/

protocol not not limited to SIP. The Internet draft [60] describes SIP usage for RELOAD.

The differences between CoSIP and P2PSIP are as follows: P2PSIP is intended to function without central entities. Discussions at the IETF where the author of this thesis was involved[8] argued for the necessity of a central authority in order to counter Sybil attacks and chosen-location attacks. Thus, the deployment of a certificate enrollment server was introduced in the RELOAD draft. Nevertheless, the functionality of this central entity has not been further specified. The intention is to keep it as simple as possible in order to save the infrastructure costs that would be otherwise required.

In contrast to P2PSIP, in CoSIP user registration and session establishment run in parallel at the server and the DHT for the following reasons:

- Unless the network is sufficiently small such as the establishment of a security context could be performed manually, a central authority is required anyway. A fundamental result on authentication protocols by Boyd [19] says that authentication can either be based on an already existing context or on a trusted third party which is a central authority.

- Session establishment does not generate a high workload on the SIP infrastructure as it is commonly believed. In his dissertation, Singh [133] showed that a cluster of six commodity PCs can support 10 million busy hour call attempts (BHCA), and 10 million users, and thus, exceeds the performance of a typical class-5 PSTN switch costing millions of dollars. Therefore, the CoSIP approach does not require significant additional resources compared to a P2PSIP solution with an enrollment server.

- In CoSIP, the SIP infrastructure is used for session establishment under normal operation provides advantages compared to a pure P2PSIP solution. First, the signaling performance using the server is in $O(1)$ while using a DHT for signaling usually results into signaling within $O(\log n)$ hops. DHTs which provides faster lookup, e.g., as proposed in this thesis in Chapter 6 are appropriate for server farms but are not likely to scale for P2P networks with endhosts. The second and even more relevant advantage of using the SIP server under normal operation is *better SPIT mitigation*. SPIT mitigation requires an extensive knowledge of the current status of the SIP networks, reputations, etc. These mechanisms can hardly be deployed in a fully distributed setup.

For these reasons, we came to the conclusion that although the IETF P2PSIP solution RELOAD in the meanwhile does not exclude the usage of a certificate enrollment server, CoSIP remains a superior approach by

[8] See, e.g., `http://www.ietf.org/mail-archive/web/p2psip/current/msg03470.html`

using the P2P network to improve the reliability of SIP signaling, though in combination with SIP servers.

Skype

Skype has become very popular in the last few years. Skype uses a mixture of client/server and P2P technologies. Some tasks, such as lookups, are performed on a P2P basis and others, such as login and PSTN gateways use the centralized servers. Being a hybrid architecture, Skype shares several commonalities with CoSIP. However, the main problem with Skype is that it uses a proprietary protocol. Our intention is to develop an open protocol for VoIP signaling which can be deployed by different vendors and service providers and interoperate with existing client-server SIP infrastructures. Furthermore, given that Skype is a closed source application, the threat model is different than in CoSIP or P2PSIP. Thus, different mechanisms for security are required in CoSIP or P2PSIP as discussed in Chapter 7.

CoDNS

DNS provides a similar functionality to SIP, since it resolves a Fully Qualified Domain Name (FQDN) to the location of a server (i.e., IP address and optionally port number). There has been several proposals for using a P2P network for decentralized DNS signaling [28, 84, 99, 109]. An extensive comparison with these approaches is out of scope for this thesis. However, we would like to mention one of them which is the most similar to CoSIP: The name "CoSIP" is actually inspired by CoDNS [99]. Using CoDNS, DNS clients send a DNS lookup to another DNS client in another domain when they notice that their DNS server is not responding. However, the authors of CoDNS do not address security issues. They admit that CoDNS can not cope with forged DNS responses. CoSIP on the other hand, can cope with the security issues by having a central authority. User contact records stored in the DHT are integrity-protected (See Chapter 7). On the other hand, CoDNS could be extended to support signed DNS records based on DNSSEC.

4.6 Conclusions

In this chapter, we presented CoSIP, which is our approach to cope with SIP reliability. CoSIP benefits from the advantages of both server-based and P2P-based SIP networking. It is more likely to survive catastrophic network failures than server-based SIP and provides better security than P2PSIP. We used Skype traces collected by Guha et al. [48] to develop a model for the reliability of CoSIP. As a side effect of our CoSIP reliability analysis, we

observed that the Skype supernode lifetime can be better modeled with a Weibull distribution than with a power-law distribution as in previous work.

CoSIP reliability models based on KAD and Skype indicate that a reliability of "3 nines" or "5 nines" can be achieved with a small overhead, such as CoSIP server downtimes can be bridged with high probability. Thus, the reliability of CoSIP is expected to be significantly higher than the reliability of a pure server solution. CoSIP provides additionally the benefit of geographic diversity of the peers (plus server) and potentially diversity in their software and hardware as well. Unless a Caller UA looses connectivity at its access network, it can with very high probability either reach the server or one of the replica nodes storing the contact data of the Callee UA.

Our prototype CoSIP implementation acts as a local SIP proxy and can be used with standard SIP clients. We successfully validated the functionality of CoSIP on local testbeds as well as on PlanetLab. Security analysis of CoSIP is provided in Chapter 7.

5. SUPERVISED NODE IDS

A CoSIP server is involved in forming the P2P network. Notably, it assigns verifiable node IDs to nodes joining the network. The distribution of nodes along the identifier space in a DHT is critical for the performance and efficiency of the DHT. DHT algorithms like Chord [144], Pastry [123], CAN [110] and Kademlia [86], propose to choose node IDs uniformly at random. It has been shown that under random node ID distribution, given n peers and m items to be stored, each peer has to store at most $O(log(n) \cdot m/n)$ *with high probability*. The term "*with high probability*" (w.h.p.) means here "with probability $\geq (1 - 1/n)$". In fact, it is likely that a few unlucky nodes will have to store $\Omega(log(n) \cdot m/n)$, which means $log(n)$ more than the average load $\lambda = {}^{m}/_{n}$.

In this chapter, we explore the benefits of a supervised node ID assignment. An *overlay supervisor* is responsible for generating node IDs when new peers enter the network. We seek a solution for node ID assignment which preserves the advantages of *consistent hashing* [65], i.e., changes have a local scope when a new node joins and leaves the network. However, the overlay supervisor guarantees a *near-optimal* node ID distribution where each node is responsible for $2/n$ of the identifier space at most; in contrast to an *optimal* node ID distribution where each node is responsible for exactly $1/n$ of the identifier space.

We provide exact formula for the load distribution $P[\#items = k]$ for:

i) the random node ID case which is commonly used in the most prominent DHT algorithms such as Chord, Pastry and Kademlia,

ii) the near-optimal case proposed in this chapter,

iii) the optimal node ID case.

Further, since the expected maximum load is an indication that the DHT is susceptible to hot spots, we estimate the expected maximum load among all peer nodes in all three cases via simulations. Our goal is to have an estimation as precise as possible of the potential improvement in load balancing in the case of near-optimal node IDs provided by an overlay supervisor, compared to the case of self-organized, and thus random node IDs.

The rest of this chapter is organized as follows. Section 5.1 provides background on consistent hashing. Section 5.2 describes the followed approach for near-optimal node IDs. Section 5.3 provides an analysis of the

load distribution $P[\#items = k]$ via simulations as well as exact formula. Section 5.4 focuses on the expected maximum load in all three cases. Section 5.5 provides an overview of related work on load balancing in DHTs. Section 5.6 concludes this chapter.

5.1 Background

Consistent hashing has been introduced in [65]. It aims for each peer receiving roughly the same number of items to be stored. It avoids the necessity of redistributing the complete hash table each time a new peer joins the network. Any change in the network, e.g., node join and leave, requires only adjustments at a local scope. Consistent hashing is generally useful where multiple machines with different views of the network need to agree on common tasks without communication. DHTs like Chord, Pastry, CAN and Kademlia use a relaxed version of consistent hashing. Instead of "k-universal hash functions" [65], cryptographic hash functions, e.g., SHA-1, are used to generate node identifiers or data item keys. Each data item is assigned to the closest peer to the item's key according to the distance metric used by the DHT. As discussed in the original paper of consistent hashing [65] and in the Chord paper [144], assuming a random node ID distribution and random keys, w.h.p. each peer has to store at most $O(\lambda \cdot log(n))$ where $\lambda = m/n$ is the average load.

In an optimal node ID distribution, the distance between two successive nodes is $1/n$. Nodes can have, e.g., the IDs k/n, $k = 0 \ldots, n-1$. We call this approach in the rest of this chapter, *optimal node IDs*. The drawback of this approach is that it would require a global agreement between all nodes, and would require changing the node IDs for all nodes every time a node joins or leaves the network.

5.2 Near-Optimal Node IDs

We aim for an approach for node ID distribution which provides better load balancing than the random IDs' approach but does not require as much coordination overhead as the optimal node IDs'. Karger and Ruhl [66] suggest such an approach. For two integers a and b, let $2b+1/2^a$ be denoted by $<a, b>$. Near-optimal node IDs are chosen from:

$$\mathcal{N} = \{<a, b> \mid a, b \in \mathbb{N}, a \geq 0, 0 \leq b < 2^{a-1}\} \tag{5.1}$$

For example,

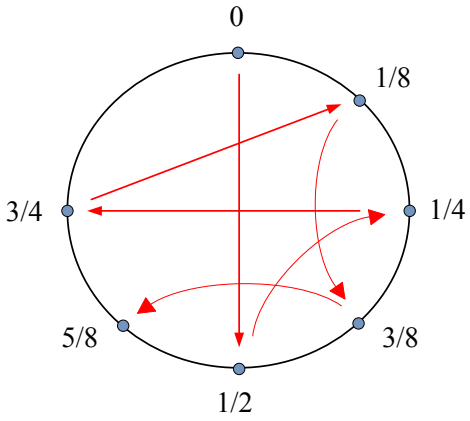


Fig. 5.1: Near-optimal node ID distribution: $0 \prec \frac{1}{2} \prec \frac{1}{4} \prec \frac{3}{4} \ldots$

- $a = 0 \quad \Rightarrow \quad 0 \leq b < 2^{-1} \quad \Rightarrow \quad b \in \{0\} \quad \Rightarrow \quad <a, b> \in \{\frac{1}{1}\} = \{1\}$
- $a = 1 \quad \Rightarrow \quad 0 \leq b < 2^{0} \quad \Rightarrow \quad b \in \{0\} \quad \Rightarrow \quad <a, b> \in \{\frac{1}{2}\}$
- $a = 2 \quad \Rightarrow \quad 0 \leq b < 2^{1} \quad \Rightarrow \quad b \in \{0, 1\} \quad \Rightarrow \quad <a, b> \in \{\frac{1}{4}, \frac{3}{4}\}$
- $a = 3 \quad \Rightarrow \quad 0 \leq b < 2^{2} \quad \Rightarrow \quad b \in \{0, 1, 2, 3\} \quad \Rightarrow \quad <a, b> \in \{\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}\}$

Now, the set $\mathcal{N}$ can be unambiguously ordered by an order relationship $\prec$ defined as follows:

$$<a, b> \quad \prec \quad <a', b'> \quad iff \quad a < a' \quad or \quad (a = a' \quad and \quad b < b') \tag{5.2}$$

This yields node IDs in the following order:

$$0 = 1 \prec \frac{1}{2} \prec \frac{1}{4} \prec \frac{3}{4} \prec \frac{1}{8} \prec \frac{3}{8} \prec \frac{5}{8} \prec \ldots \tag{5.3}$$

Figure 5.1 shows the placement of nodes in this order. We call this node ID distribution a *near-optimal* node ID distribution in contrast to an *optimal* node ID distribution with equidistant successive nodes IDs. The ratio between the minimum and the maximum distance between two successive nodes in a near-optimal node ID distribution is limited by the factor of two.

Karger and Ruhl [66] denote such a node ID distribution as an *Ideal State*. They aim for approximating such an Ideal State in a decentralized environment by reactively relocating nodes when necessary. Instead of a decentralized approach, we take a centralized approach with an *overlay supervisor*. The supervisor assigns node IDs to new coming peers according to the Ideal State.

Note that the load imposed on nodes which have to store more items is not only storage load. They potentially have to answer more lookup queries.

However, this depends on the distribution of the popularity of the items in the DHT. Our load analysis is purely based on the number of items stored at each peer as a load metric.

Note also that the load can be modeled with the "Bins and balls" problem where m balls are sequentially thrown into a bin out of n bins chosen independently and uniformly at random. The case of equal sized bins (which is equivalent to the optimal node ID distribution here), has been extensively analyzed in the literature. We use the latest results on this topic [107] to validate our simulation results.

5.3 Load Distribution

In this section, we provide an analysis of the load distribution via simulations and mathematical models for the three different cases for node assignment:

i) random IDs,

ii) near-optimal node IDs,

iii) optimal node IDs.

5.3.1 Simulations

We simulated a network with $n = 10^4$ nodes and $m = 5 \cdot 10^5$ items to be stored. Thus, the average (and ideal) number of items per node is $\lambda = {}^m/_n = 50$. Each item is uniquely assigned to its successor node in the ring $[0, 1)$ (no replication). Note that we chose the same parameters for n and m as in the Chord paper [144], which allows for a direct comparison. Figures 5.2, 5.3 and 5.4 show the probability density function (PDF) of the number of items per nodes with uniformly distributed random node IDs, with near-optimal node IDs and with an optimal node ID distribution. As a point of reference, we added respectively the line representing the ideal distribution where all nodes would have an equal load of λ items. As expected, the load in the optimal and near-optimal node ID cases is closer to the mean value $\lambda = 50$, such as most of the peers have to store $< 2 \cdot \lambda$ items. The random node ID distribution creates a skewed load distribution with many nodes having to store more than $> 4 \cdot \lambda$. Also as expected, Figure 5.2 shows the same behavior in the Chord paper [144].

5.3.2 Mathematical Model

After these preliminary simulation runs, we are interested in the mathematical model behind these figures. We assume a continuous ID space ID $[0, 1)$. Further, let $B_{(m,p)}$ denote the binomial distribution with success probability

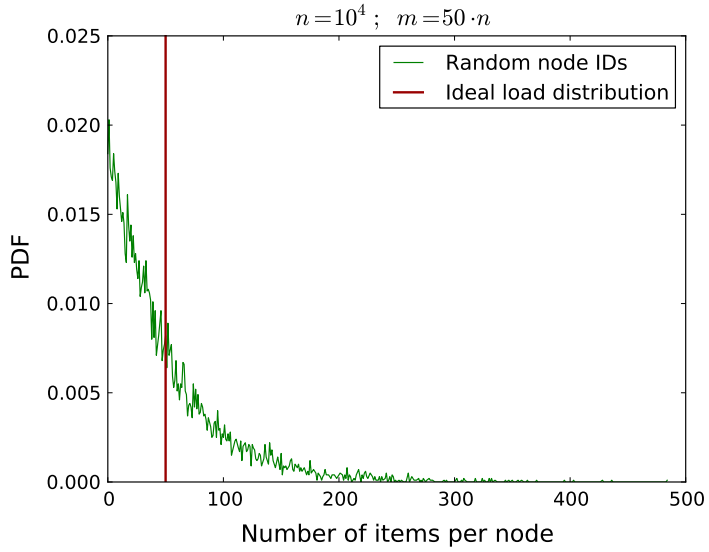


Fig. 5.2: The PDF of the number of items per node given random node IDs.

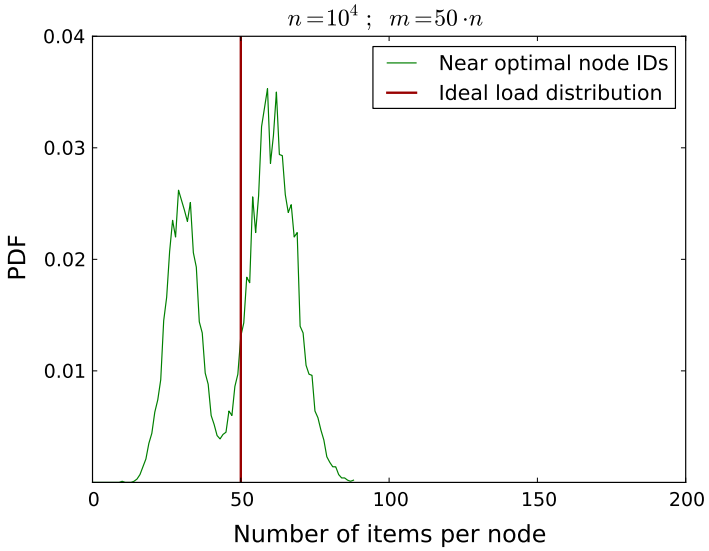


Fig. 5.3: The PDF of the number of items per node given near-optimal node IDs.

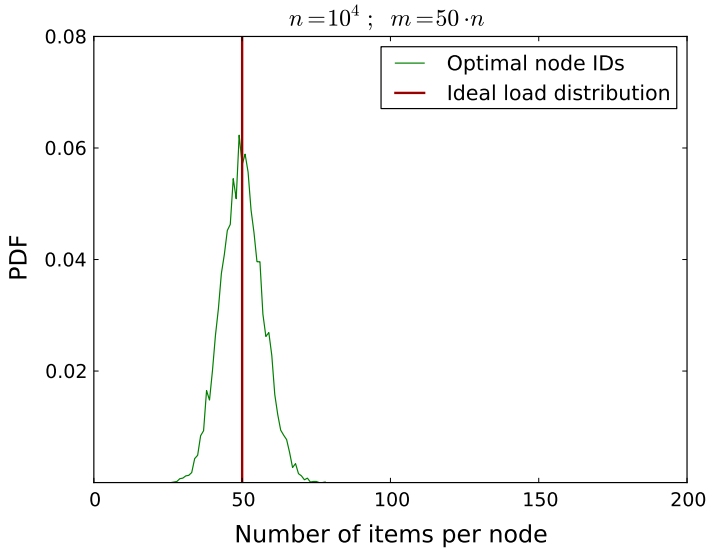


Fig. 5.4: The PDF of the number of items per node given optimal node IDs.

ρ and number of trials m.

$$B_{(m,\rho)}(k) = \binom{m}{k} \rho^k (1-\rho)^{m-k} \tag{5.4}$$

5.3.2.1 The Optimal Node IDs Case

We start with the simplest case, which is the case of an optimal node ID distribution (Figure 5.4). Under the assumption that the item keys are random and independent, the probability whether a certain item is stored at a certain node can be modeled as a Bernoulli experiment with success probability $1/n$. Thus, the probability that a node will store k items is given by:

$$P[\#items = k] = B_{(m,\frac{1}{n})}(k) \tag{5.5}$$

5.3.2.2 The Near-Optimal Node IDs Case

For the near-optimal node ID distribution, then two cases can be differentiated:

- $\exists\, p \in \mathbb{N}, n = 2^p$. This case is equivalent to the optimal load distribution.
- $n \neq 2^p \quad \forall p \in \mathbb{N}$. In this case, let $p = \lfloor log_2(n) \rfloor$. Thus, p is the greatest integer that fulfills $2^p \leq n$.

In the second case, the distance between two successive nodes varies up to the factor of two. Let I_1 be the interval between the node with ID 0 and its successor, and I_2 the interval between 0 and its predecessor (See Figure 5.1). Thus, by the definition of the node IDs, $|I_2| = 2 \cdot |I_1|$. Let n_1 be the number of successive nodes which are responsible for an interval with length $|I_1|$ (e.g., nodes with IDs 1/8, 1/4 and 3/8, 1/2, 5/8 and 3/4 in Figure 5.1)) and n_2 the number of successive nodes which are responsible for an interval with $|I_2|$ (e.g., node with IDs 0 in Figure 5.1).

In order to determine n_1 and n_2, we need to go one step back in the overlay construction procedure. When nodes join the network successively, at a certain point of time there are exactly 2^p nodes in the network. All nodes are responsible for an interval which length is $|I_2|$. Then $(n - 2^p)$ new nodes join the network successively. Each of them splits one interval with length $|I_2|$ to two intervals with length $|I_1|$. By the time the number of nodes increases to n, the new $(n - 2^p)$ nodes have split $(n - 2^p)$ intervals with length $|I_2|$ to $n_1 = 2 \cdot (n - 2^p)$ intervals with length $|I_1|$.

Now, we can unambiguously determine n_2 as well:

$$\begin{aligned} n_2 &= n - n_1 \\ &= n - 2(n - 2^p) = n - 2n + 2^{p+1} = n - 2n + 2^{p+1} \\ &= 2^{p+1} - n \end{aligned} \tag{5.6}$$

Then, we can determine $|I_1|$ and $|I_2|$ as follows:

$$1 = n_1 \cdot |I_1| + n_2 \cdot |I_2| \tag{5.7}$$

$$= 2(n - 2^p) \cdot |I_1| + (2^{p+1} - n) \cdot 2|I_1| \tag{5.8}$$

$$= (2n - 2^{p+1} + 2^{p+2} - 2n) \cdot |I_1| \tag{5.9}$$

$$= 2^{p+1} \cdot |I_1| \tag{5.10}$$

Thus,

$$|I_1| = \frac{1}{2^{p+1}} \tag{5.11}$$

$$|I_2| = \frac{1}{2^p} \tag{5.12}$$

Given an arbitrary node responsible for an interval I, then either $|I| = |I_1|$ or $|I| = |I_2|$. Thus,

$$\begin{aligned} P[\#items = k] &= P[\ \#items = k\ \wedge\ P[|I| = |I_1|\ ] \\ &+ P[\ \#items = k\ \wedge\ P[|I| = |I_2|\ ] \end{aligned} \tag{5.13}$$

Since

$$P[\#items = k \mid |I| = |I_1|\,] = \frac{P[\,\#items = k \;\wedge\; |I| = |I_1|\,]}{P[\,|I| = |I_1|\,]} \tag{5.14}$$

we can deduce that

$$\begin{aligned} P[\,\#items = k \;\wedge\; |I| = |I_1|\,] &= P[\,|I| = |I_1|\,] \cdot P[\#items = k \mid |I| = |I_1|\,] \\ &= \frac{n_1}{n} \cdot P[\#items = k \mid |I| = |I_1|\,] \end{aligned} \tag{5.15}$$

On the other hand, $P[\#items = k \mid |I| = |I_1|\,]$ follows a binomial distribution $B_{(m,|I_1|)}$ with number of trials m and success probability $|I_1|$ at each trial. Thus,

$$\begin{aligned} P[\,\#items = k \;\wedge\; |I| = |I_1|\,] &= \frac{n_1}{n} \cdot B_{(m,|I_1|)}(k) \\ &= \frac{n_1}{n} \cdot B_{(m,\frac{1}{2^{p+1}})}(k) \end{aligned} \tag{5.16}$$

Likewise for the 2nd summand of equation (5.13)

$$\begin{aligned} P[\,\#items = k \;\wedge\; |I| = |I_2|\,] &= \frac{n_2}{n} \cdot B_{(m,|I_2|)}(k) \\ &= \frac{n_2}{n} \cdot B_{(m,\frac{1}{2^p})}(k) \end{aligned} \tag{5.17}$$

This results into the overall probability $P[\#items = k]$ being a wighted sum of two binomial distributions:

$$P[\#items = k] = \frac{n_1}{n} \cdot B_{(m,\frac{1}{2^{p+1}})}(k) + \frac{n_2}{n} \cdot B_{(m,\frac{1}{2^p})}(k) \tag{5.18}$$

Figure 5.5 shows the simulation as well as the formal model of a DHT with a near-optimal ID distribution with $n = 10^4$ nodes and $m = 5 \cdot 10^5$ items.

5.3.2.3 The Random Node IDs Case

Now we move further to the random node ID case. Let I be the interval that an arbitrary node is responsible for and $|I|$ the length of that interval. As in Section 5.3.2.2, we first explore the different values that $|I|$ can take and deduce the overall probability $P[\#items = k]$. In contrast to the near-optimal node ID distribution case, $|I|$ can be any value in $[0, 1)$ and not only $|I_1|$ or $|I_2|$.

Thus,

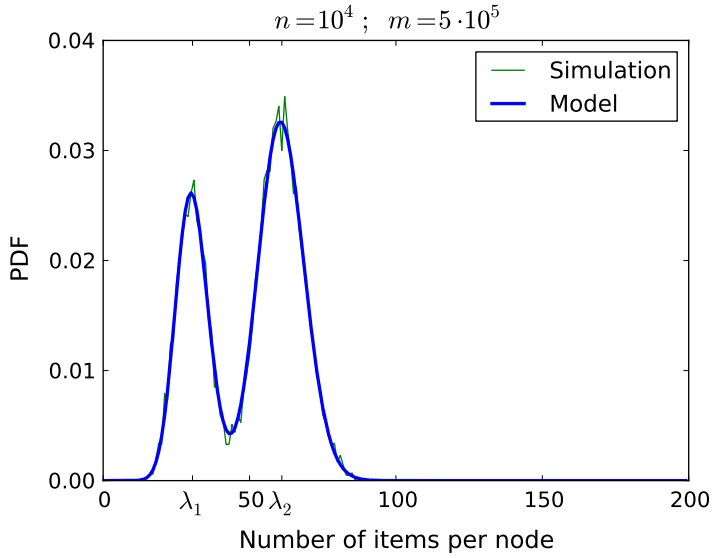


Fig. 5.5: Comparison of simulation and formal model of the PDF of the number of items per node given near-optimal node IDs. Number of nodes $n = 10^4$. Number of total items $m = 5 \cdot 10^5$.

$$P[\#items = k] = \int_{[0,1)} P[\ \#items = k\ \wedge\ P[|I| = x\ ]\ dx$$

Let $l(x)$ be the probability density function of $|I|$. Thus,

$$P[\#items = k] = \int_0^1 l(x) \cdot P[\#items = k \mid |I| = x\ ]\ dx \tag{5.19}$$

Lemma 5.1: *The probability density function $l(x)$ of the interval length an arbitrary node is responsible for is*

$$l(x) \ = \ \begin{cases} (n-1)(1-x)^{n-2} & 0 \leq x < 1 \\ \\ 0 & otherwise \end{cases} \tag{5.20}$$

Proof: Let $L(x)$ be the cumulative distribution function of $|I|$. Thus,

$$L(x) = P[|I| \leq x] \tag{5.21}$$

and

$$\frac{dL(x)}{dx} = l(x) \tag{5.22}$$

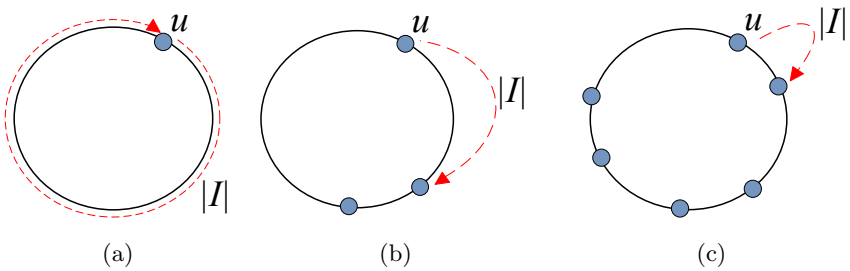


Fig. 5.6: Interval length $|I|$ as minimum of clockwise distance from u to other nodes.

For the statistical analysis, it does not matter if the node at the beginning or the end of an interval is responsible for the data. Thus, we assume that items are uniquely assigned to the predecessor node. As shown in Figure 5.6, for an arbitrary node u $|I|$ is the clockwise distance from u to the next node. Let $L_i(x)$, $i = 1, \cdots, n-1$ be the cumulative distribution functions modeling the distance between u and an arbitrary other node. Thus,

$$L = min\{L_1, L_2, L_{n-1}\} \tag{5.23}$$

Given uniformly distributed random node IDs, $L_i(x)$ follows the continuous uniform distribution:

$$L_i(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad 0 \leq x < 1 \\ 1 & \text{for} \quad x \geq 1 \end{cases} \tag{5.24}$$

Thus, $L(x)$ is the cumulative distribution of the minimum of $(n-1)$ independent random variables, each of them describing the clockwise distance from an arbitrary node u to one of the other $(n-1)$ nodes. Thus,

$$L(x) = 1 - \prod_{i=1}^{n-1} (1 - L_i(x)) \tag{5.25}$$

This results into:

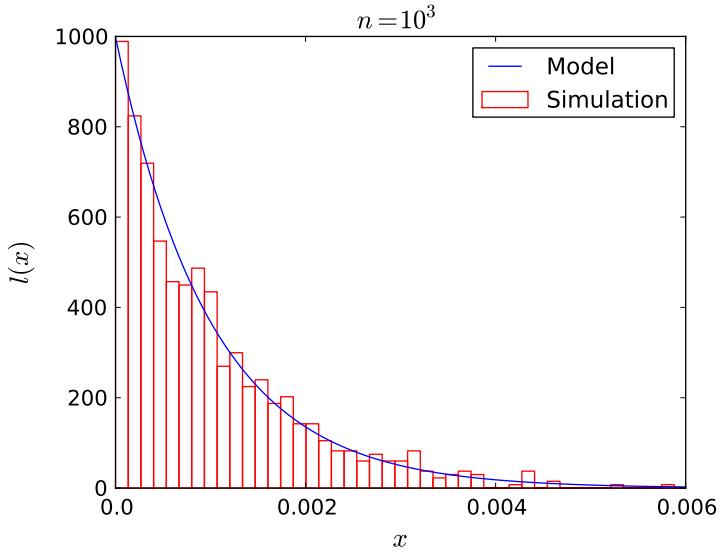


Fig. 5.7: PDF of the portion of the address space that an arbitrary node is responsible for under random node IDs.

$$L(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 - (1-x)^{n-1} & \text{for} \quad 0 \leq x < 1 \\ 1 & \text{for} \quad x \geq 1 \end{cases} \tag{5.26}$$

Thus,

$$l(x) = \begin{cases} (n-1)(1-x)^{n-2} & 0 \leq x < 1 \\ 0 & \textit{otherwise} \end{cases} \tag{5.27}$$

∎

With a quick simulation, we can visualize the result of Lemma 5.1 as shown in Figure 5.7.

Now, we can proceed with the load distribution for the random node ID

case. From equation 5.19, and Lemma 5.1 we can deduce that:

$$\begin{aligned} P[\#items = k] &= \int_0^1 (n-1)(1-x)^{n-2} \cdot B_{(m,x)}(k)\, dx \\ &= \int_0^1 (n-1)(1-x)^{n-2} \cdot \binom{m}{k} x^k (1-x)^{m-k}\, dx \\ &= (n-1)\binom{m}{k} \int_0^1 x^k \cdot (1-x)^{m+n-k-2}\, dx \end{aligned} \tag{5.28}$$

For a simpler presentation, let $s = m + n - k - 2$. Thus,

$$P[\#items = k] = (n-1)\binom{m}{k} \int_0^1 x^k \cdot (1-x)^s\, dx \tag{5.29}$$

The term $(1-x)^s$ can be expanded into $\sum_{l=0}^{s} \binom{s}{l}(-x)^l$. Thus,

$$\begin{aligned} P[\#items = k] &= (n-1)\binom{m}{k} \int_0^1 x^k \cdot \sum_{l=0}^{s} \binom{s}{l}(-x)^l\, dx \\ &= (n-1)\binom{m}{k} \sum_{l=0}^{s} (-1)^l \int_0^1 x^{k+l} dx \\ &= (n-1)\binom{m}{k} \sum_{l=0}^{s} (-1)^l \left[\frac{1}{k+l+1}\, x^{k+l+1} \right]_0^1 \\ &= (n-1)\binom{m}{k} \sum_{l=0}^{s} \frac{(-1)^l}{k+l+1} \end{aligned} \tag{5.30}$$

Figure 5.8 shows the simulation results together with the formal model.

5.4 Maximum Load Distribution

In the previous section, we provided simulations as well as exact formula for the load distribution. These results are already helpful to estimate the benefits of near-optimal node IDs compared to random node IDs. However, the load distribution is not sufficient for an adequate estimation of the expected maximum load. Particularly, it is not sufficiently significant to derive

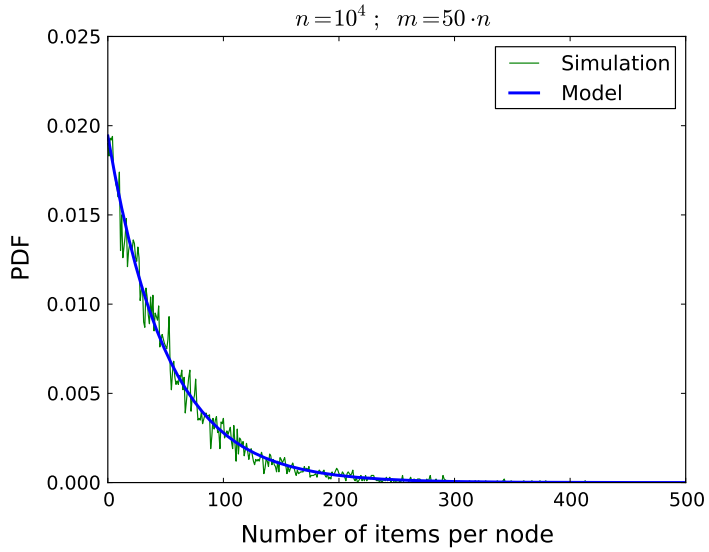


Fig. 5.8: Comparison of simulation and formal model of the PDF of the number of items per node given random node IDs.

a statement whether the DHT is susceptible to hot spots. We are notably interested whether there are some nodes with a load in $\Omega(\lambda \cdot log(n))$, which means $log(n)$ times more than the average load.

Let M be the random variable that count the maximum number of items assigned to any node in the P2P network, i.e.,

$$M = max\{\#items\ assigned\ to\ u \mid u \in V\} \tag{5.31}$$

where V is the set of nodes in the network. In case of uniformly random node IDs, we denote M by M_{random}. In case of near-optimal node IDs or optimal node IDs, we denote M by $M_{near-optimal}$ or $M_{optimal}$ respectively.

In this section, we provide estimations of $E(M_{random})$, $E(M_{near-optimal})$ and $E(M_{optimal})$ via simulations. In the optimal node ID case, we compare our simulation results with the model provided by Raab and Steger for the bins and balls problems [107]. Further, we deduce an upper bound and lower bound for the near-optimal case $E(M_{near-optimal})$.

5.4.1 Simulation Setup

We first consider an average load $\lambda = 100$ and vary the number of nodes in the network $n \in \{10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$. For each value of n, we perform a simulation with n nodes and $m = \lambda \cdot n$ items and measure the maximum load. We repeat this experiment 500 times for each value of n. Figure 5.9 shows the mean values of the maximum number of items per node over the 500 experiments, together with the 10% and 90% percentiles. Note

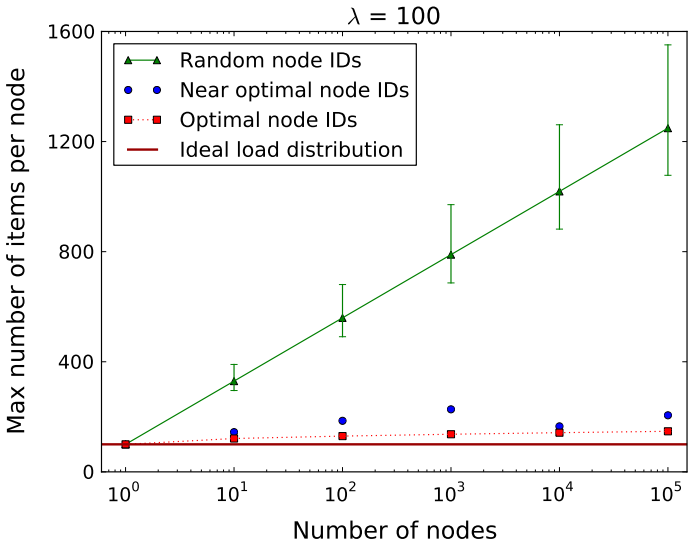


Fig. 5.9: Mean and 10%, 90% percentiles of the maximum load for different values of n and a constant $\lambda = 100$. Number of simulation runs: 500

that the difference between the mean values and 10% and 90% percentiles is quite visible for the random node ID case, while it is not visible for the optimum and near optimum node ID case since it is quite small (≤ 11 items in our experiments).

In the second experiment, we consider a constant number of nodes $n = 10^3$ and vary the average load λ from 0 to 100 in additive increments of 20. For each value of λ we measure the maximum load and repeat the experiment 500 times. Figure 5.10 shows the mean values of the maximum number of items per node over the 500 experiments, together with the 10% and 90% percentiles.

5.4.2 Interpretation

Random Node IDs

In Figure 5.9, we can see that for the random node ID case, the maximum grows logarithmically with n. Using regression analysis, we determine the slope and the intercept of the measured values

$$\hat{E}_{random}(M) = \alpha \cdot log(n) + \beta \tag{5.32}$$

with a 95%-confidence interval. The result is $\alpha = 99.77 \pm 4.10$, and $\beta = 75.75 \pm 28.56$. Given that $\lambda = 100$, the computed value of α is an indication that

$$E(M_{random}) = \lambda \cdot log(n) + \beta \tag{5.33}$$

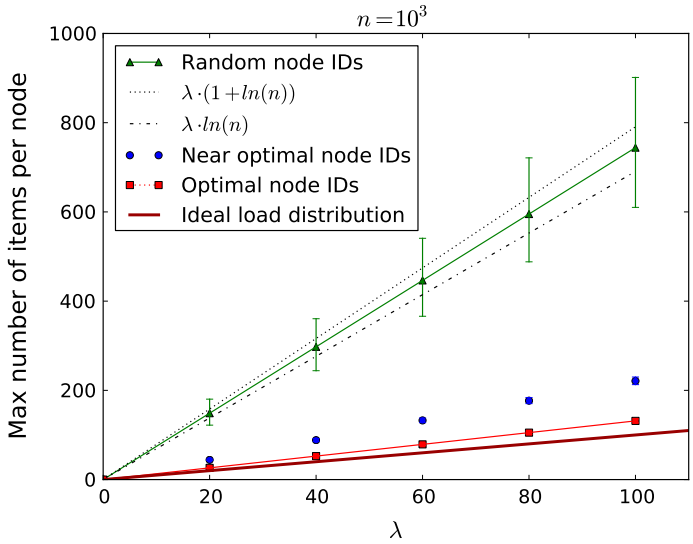


Fig. 5.10: Mean and 10%, 90% percentiles of the maximum load for different values of *lambda* and a constant $n = 1000$. Number of simulation runs: 500

In fact, as it can be observed in Figure 5.10 $E_{random}(M)$ is between $\lambda \cdot log(n)$ and $\lambda \cdot (1 + log(n))$.

Optimal Node IDs

In Figure 5.9, we can observe that the maximum load for the optimal node ID case grows with n as well. This is conform to the findings of Raab and Steger on the bins and balls problem [107].

Further, it can be observed that the maximum load increases much less than in the random node ID case.

Near-Optimal Node IDs

As for the near-optimal node ID case, the maximum load depends on the offset of n to the next smaller power of two, i.e., 2^p with $p = \lfloor log_2(n) \rfloor$. This explains why it is not monotonic by n in Figure 5.9.

As we explained in Section 5.3, there are

- $n_1 = 2 \cdot (n - 2^p)$ intervals with length $|I_1| = 1/2^{p+1}$
- $n_2 = (2^{p+1} - n)$ intervals with length $|I_2| = 1/2^p$

Obviously, the maximum load is expected to fall into one of the nodes with the greater interval length $|I_2|$.

$$E(M_{near-optimal}(n,\lambda)) = E(M_{optimal}(n_2,\lambda_2)) \tag{5.34}$$

λ_2 has an upper bound which is 2λ given that:

$$\frac{\lambda_2}{\lambda} = \frac{\frac{m}{2^p}}{\frac{m}{n}} = \frac{n}{2^p} \leq \frac{2^{p+1}-1}{2^p} = 2 - \frac{1}{2^p} < 2 \tag{5.35}$$

Thus, $n_2 \leq n$ and $\lambda_2 < 2\lambda$.

According to the simulations, $E(M_{optimal}(n,\lambda))$ is monotonically increasing by n as well as λ. Thus,

$$E(M_{near-optimal}(n,\lambda)) \leq E(M_{optimal}(n,2\lambda)) \tag{5.36}$$

On the other hand,

$$E(M_{optimal}(n,\lambda)) \leq E(M_{near-optimal}(n,\lambda)) \tag{5.37}$$

Equations 5.36 and 5.37 provide upper and lower bounds for $E(M_{near-optimal})(M(n,\lambda))$.

Unfortunately, it has not been possible to provide a closed formula for $E(M_{random})$, $E(M_{near-optimal})$ and $E(M_{optimal})$[1]. However, the simulation results shown in Figure 5.9 and Figure 5.10 show that there is a clear benefit from using near-optimal IDs compared to random IDs.

5.5 Related Work

There has been a good deal of work to optimize the load in DHTs. The mechanisms can be classified into the two following categories:

- *address-based* approaches where nodes are organized in the networks such as each node is responsible for $O(1/n)$ of the address space,
- *item-based* approaches where either the nodes or the items to be stored are organized in the networks such each node is responsible for $O(1/n)$ of the number of items. These mechanisms can partially be extended to cope with different load metrics, i.e., considering, for example, the different popularity of items.

[1] The results of Raab and Steger [107] show that this is quite challenging even for the optimal case $E(M_{optimal})$.

The approach proposed in this chapter is thus an address-based approach. However, it can be clearly differentiated from previous approaches in two aspects:

- It is a *supervised* approach. Other approaches require decentralized coordination between the peers and often a high maintenance overhead to re-distribute the load (or the nodes along the address space). Thus, other approaches require the nodes to be trustworthy and cooperative and, e.g., willing to accept load from other peers. This is rarely the case, particularly if the P2P networks consists of non trustworthy endhosts. Even worse, one has to cope with chosen-location attacks (See Section 3.3.8) in such setup; thus nodes should not be allowed to arbitrarily choose their IDs.

- Our approach is *proactive* while other approaches are *reactive*. In our approach, nodes are organized in the network from the beginning such as each node is responsible for $O(1/n)$ of the address space. Other approaches require a periodic assessment of the load in the network in order to react accordingly.

Among the reactive approaches, one can also differentiate between *active* and *passive* mechanisms.

- Passive mechanisms are triggered only when a change occurs in the network, e.g., a node joins or leaves, or an item is added or removed.

- Using active mechanisms, nodes actively probe the network periodically to search for nodes to balance.

Among the different approaches categorized above, we discuss two representative approaches from the literature. namely, the *Power-of-Two Choices* [23] and *Virtual Servers* [67].

The Power-of-Two Choices approach optimizes load when inserting new elements, thus is clearly a passive approach. It distributes the load by storing an item at the least loaded place out of a set of candidate places. This works by applying multiple, say k, hash functions $h_1, h_2, ..., h_k$ for determining k candidates. An item with a name 's' can now be stored on any of the k nodes that is responsible for one of the IDs $h_1(s)$, $h_2(s)$, ..., $h_k(s)$. Then, the node with the least load is chosen to store the item. As a consequence, lookup requests need to be sent to all k nodes. Alternatively, only one node of the k nodes stores the item, and the remaining $(k-1)$ nodes store pointers to the node storing the item. For the case $k = 2$, there are always two nodes to be chosen from. This explains the name of this approach "power-of-two choice".

The Virtual Servers (VS) approach [67] propose that each "physical" node participates in the network at $O(log\, n)$ positions by running a virtual node on each position. Nodes react to overload by checking their status

periodically and transferring load among each other. Thus, VS is a representative for active approaches. The drawbacks of VS approach are manifold. First, nodes need to keep routing state for each virtual node. Thus, VS require a high maintenance overhead. Second, this maintenance overhead is extrapolated by churn if data items have to be shifted between nodes upon joins and leaves. Third, it is assumed that nodes would have the incentives to cooperate with each other to share the load.

The near-optimal node distribution described in this chapter has been suggested by Karger and Ruhl [66]. They provide a reactive approach for approximating such a near-optimal node distribution in a purely decentralized P2P network. Like the Virtual Servers approach, they propose to keep the state for $O(log\, n)$ different virtual nodes for each physical node. Then, nodes must change their position to adapt the current node ID distribution and approximate a near-optimal node ID distribution. Then, they prove that their approach improves consistent hashing in that every node is responsible for a $O(1/n)$ fraction of the identifier space w.h.p. The drawbacks of this approach are again that nodes need to keep state for $O(log\, n)$ routing tables and that it is assumed that nodes are willing to cooperate to share the load among each other. Moreover, the authors do no provide an analysis of the impact of such a node distribution on the actual load distribution or the expected maximum load distribution.

5.6 Conclusions

The contributions of this chapter are threefold:

Supervised Near-Optimal IDs as an Approach for Load Balancing
We investigate the potential benefits of assigning node IDs by an overlay supervisor. The supervisor keeps an overview of the node ID distribution and assigns node IDs to new joining nodes according to a simple though efficient algorithm. This algorithm allows for a near-optimal node ID distribution where each node is responsible for at most $2/n$ of the identifier space. The approach has been proposed by Karger and Ruhl [66]. But instead of a supervised approach, they provide a decentralized approach where nodes would have to switch between different virtual nodes in order to approximate the anticipated distribution of the ID space.

Load Distribution
We provided *exact formula* for the load distribution for

i) the random node ID case, which is the approach used by prominent DHT algorithms such as Chord, Pastry and Kademlia

ii) the near-optimal node ID proposed by Karger and Ruhl.

iii) and the optimal node ID case with equidistant successive nodes.

Maximum Load Distribution: While exact formula for the expected maximum load $E(M)$ in all three cases is still subject to future work and potentially quite complex mathematical analysis, our simulations indicate the clear benefit of using near-optimal node IDs over random IDs.

6. LOW-DIAMETER OVERLAYS

The most prominent DHT algorithms, such as Chord, Pastry and Kademlia keep the routing table size in $O(log\, n)$ where n is the number of nodes in the network and achieve an average path length of $O(log\, n)$. However, $O(log\, n)$ path length is not an appropriate one-size-fits-all for all applications, e.g., for mega-scale server farms with very low lookup latency as a requirement. In this chapter, we propose an overlay topology which can achieve a path length of two hops with a probability $(1 - \epsilon)$ for an arbitrarily small ϵ for random node IDs and with a probability 1 for supervised node IDs. The topology is an approximation of Generalized HyperCubes (GHC). The routing table size per node is $O(\sqrt{n})$. We present an analytical model of the proposed topology to estimate the lookup performance and validate the model through simulations.

6.1 Background

First, we provide background information, specifically on hypercubes and generalized hypercubes.

6.1.1 Optimal-Diameter Graphs

It is well known [47] that the hop count diameter D of a graph with n nodes, and a degreedeg has a lower bound:

$$D \geq \left\lfloor \frac{log(n)}{log(deg - 1)} \right\rfloor - 1 \tag{6.1}$$

For large n values, $\lfloor {}^{log(n)}/_{log(deg-1)} \rfloor - 1$ can be approximated by ${}^{log(n)}/_{log(deg)}$ and it can be deduced from equation (6.2) that

$$deg \geq \sqrt[D]{n} \tag{6.2}$$

For example, if the goal is to achieve routing with two hops, then $D = 2$, thus,

$$deg \geq \sqrt{n} \tag{6.3}$$

Equation 6.2 indicates that a diameter D requires a routing table size in $O(\sqrt{n})$. Thus, in the next section, we will see how we can build such a network.

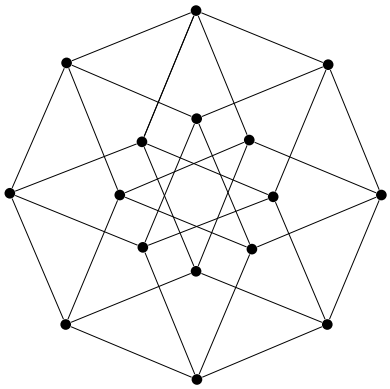


Fig. 6.1: A 4-dimensional hypercube; $n = 2^4$.

6.1.2 Hypercubes

In the rest of this chapter, we denote the set $\{0, 1, \ldots, x-1\}$ by $[x]$. Hypercubes are a generalization of squares (dimension $dim = 2$) and cubes ($dim = 3$). Figure 6.1 shows a hypercube with dimension $dim = 4$. The set of vertices (nodes) of a hypercube with dimension dim is $V = [2]^{dim}$. The number of nodes is $n = |V| = 2^{dim}$. Each node a in a hypercube is identified by a binary address $(a_{dim-1}, \ldots, a_1, a_0)$; $a_i \in [2] \quad \forall i \in [dim]$. There are different topologies, called hypercubic networks, which can result from a hypercube depending on the set of links between nodes. For example, in a mesh topology, each node is connected to neighboring nodes on each axis. Thus, the set of edges is defined by:

$$E = \{ \{(a_{dim-1}, \ldots, a_0), (b_{dim-1}, \ldots, b_0)\} | a_i, b_i \in [2], \sum_{i=0}^{dim-1} |a_i - b_i| = 1\} \tag{6.4}$$

Other classes of hypercubic networks are tori, butterflies, cube-conected-cycles, shuffle-exchange and de Bruijn networks.

6.1.3 Generalized Hypercubes

We consider a class of networks which were referred to as "Generalized Hypercubes" (GHC) in [15]. Figure 6.2 shows, e.g., a GHC with base $m = 5$ and dimension $dim = 2$. The set of vertices (nodes) of a hypercube with the base m and dimension dim is $V = [m]^{dim}$. The number of nodes is $n = |V| = m^{dim}$. Each node a in a hypercube is identified by a base-m address $(a_{dim-1}, \ldots, a_1, a_0)$; $a_i \in [m] \quad \forall i \in [dim]$.

A node in a particular axis i in a GHC is connected to all other nodes

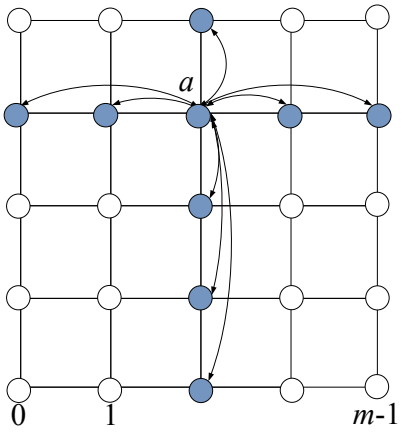


Fig. 6.2: Two-dimensional ($dim = 2$) Generalized Hypercube with $m = 5$. Only the contacts of a node a are shown for clarity.

on the same axis. Thus, the set of edges is defined by:

$$E = \{ \{(a_{dim-1}, \ldots, a_0), (b_{dim-1}, \ldots, b_0)\} | \\ a_i, b_i \in [m], \exists i \in [dim] \quad a_i = b_i\} \tag{6.5}$$

Vertices on the same axis $i \in [dim]$ build a clique. This is illustrated in Figure 6.2 although only the contacts of a node a are shown for clarity. In the rest of this chapter, we denote such a network by GHC_m^{dim}.

Routing

The routing in GHC_m^{dim} from a node a to a node b is performed by considering the digits (coordinates) of a and b which are different. The number of different digits between a and b is the *Hamming distance* between a and b. At each routing step, the Hamming distance is reduced by one digit. Let k be the Hamming distance between a and b. Then, there exist $k!$ different shortest paths with length k between a and b (all different permutations of the different positions of the digits where a and b differ).

Diameter and Degree

The diameter of GHC_m^{dim} is equal to the maximum Hamming distance between two different nodes. Thus, the diameter of GHC_m^{dim} is equal to the dimension: $D = dim$. The node degree in GHC_m^{dim} is constant for all nodes:

$$\forall a \in H_m^{dim}, \quad deg(a) = dim.m = dim. \sqrt[dim]{n} \tag{6.6}$$

6.2 Overlay Algorithm

In this section, we derive an appropriate definition of an overlay topology which approximates the topology of a two-dimensional GHC. The goal is to preserve the benefit of GHC_m^{dim} to route messages within dim hops at most. We aim at routing within two hops, thus $dim = 2$. First, we introduce the network parameters and some notation that we need for the rest of this chapter. We describe the algorithms for building routing tables and for routing lookup queries.

6.2.1 Network Parameters and Notation

Let n be the number of peers. The target value for the path length between two arbitrary nodes. $dim = 2$ is a system-wide parameter that needs to be agreed on by all participating nodes. The number of nodes in the overlay n needs to be estimated by each node separately. All peers are identified by a pair of real numbers $a = (a_1, a_0)$, where $a_i \in [0, 1)$. Thus, $V \subseteq [0, 1)^2$.

Let $m = \lceil \sqrt{n} \rceil$. m is equivalent to the basis in the approximated GHC_m^2.

Nodes are logically ordered in rings, according to their positions on each axis $i \in \{0, 1\}$. For a node u, an axis $i \in \{0, 1\}$ and a subset $U \subseteq V$ with cardinality ≥ 2, we denote by $pred_i(u, U)$ the predecessor of u in U on the ring on axis i, i.e., the node $v \in U$ such that

$$v_i = \begin{cases} max\{w_i | w \in U \quad w_i < u_i\} & if \quad \{w | w \in U \quad w_i < u_i\} \neq \varnothing \\ \\ max\{w_i | w \in U\} & otherwise \end{cases} \tag{6.7}$$

Likewise, we denote by $succ_i(u, U)$ the successor of u in U on the ring on axe i, i.e., the node $v \in U$ such that

$$v_i = \begin{cases} min\{w_i | w \in U \quad w_i > u_i\} & if \quad \{w | w \in U \quad w_i > u_i\} \neq \varnothing \\ \\ min\{w_i | w \in U\} & otherwise \end{cases} \tag{6.8}$$

Further, we denote $pred_i(u, V)$ simply by $pred_i(u)$ and $succ_i(u, V)$ by $succ_i(u)$. Finally, we denote the iterated $pred_i$ by $pred_i^{(.)}$, i.e., $pred_i^{(0)}(u, U) = u$, and $pred_i^{(l+1)}(u, U) = pred_i(pred_i^{(l)}(u, U), U)$ for all $l \geq 0$. Likewise, $succ_i^{(.)}$ the iterated successor on the axis i.

6.2.2 Overlay Routing Tables

We now explain how each peer builds its routing table in a decentralized environment. In the topology given in Figure 6.2, a peer a has its contacts in the row or column where a is positioned. The row and the column of a contain m peers respectively. The row is the set of contacts of node a

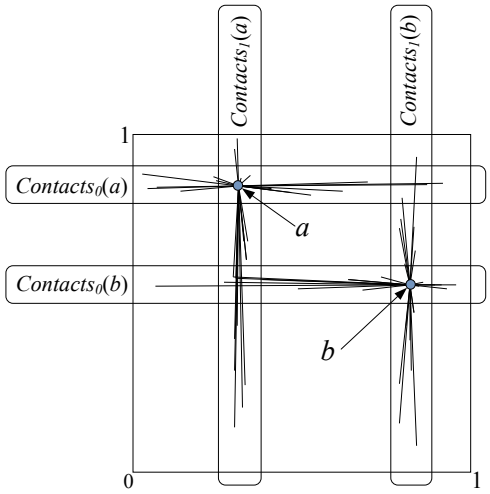


Fig. 6.3: Two nodes a and b with their links to their contacts in a network with $n = 256$

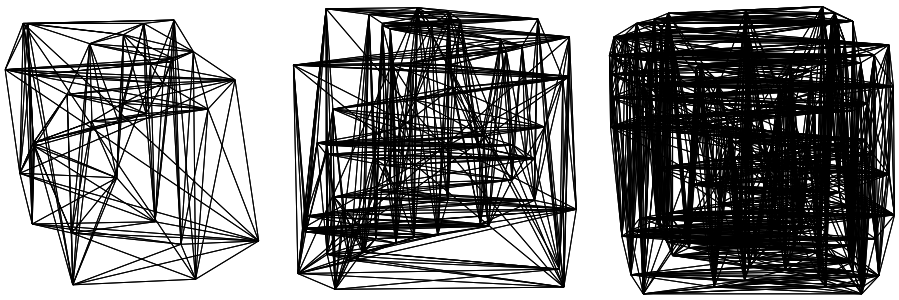
on the axe 0, which we denote by $Contacts_0(a)$. The column is the set of contacts of node a on the axe 1, which we denote by $Contacts_1(a)$. In order to approximate the row of a two-dimensional GHC, a finds m contact peers which are its immediate neighbors on the 1-axis. These can be unambiguously determined by:

$$\begin{aligned} Contacts_0(a) &= \{pred_1^{(l)}(a) \,|\, 0 \leq l \leq \left\lceil \frac{m}{2} \right\rceil\} \\ &\cup \{succ_1^{(l)}(a) \,|\, 0 \leq l \leq \left\lceil \frac{m}{2} \right\rceil\} \end{aligned} \tag{6.9}$$

In other words, $Contacts_0(a)$ is the set of $\lceil m/2 \rceil$ clockwise and $\lceil m/2 \rceil$ counter-clockwise neighbors of a along the ring on the 1-axis. The cardinality of $Contacts_0(a)$ is either m or $(m + 1)$ depending on whether m is odd or even. However, this is not relevant for the routing procedure or for our analytical model below.

As we observe in Figure 6.3, while nodes in the "row" $Contacts_0(a)$ lie closely together on the 1-axis, their coordinates on the 0-axis can be considered as a random selection of m elements from $\{u_0 | u \in V\}$. Thus, nodes in $Contacts_0(a)$ are uniformly distributed at random along the 0-axis. Figure 6.3 shows two nodes a and b with their links in a simulated network with $n = 256$ peers, thus $m = 16$.

For $Contacts_1(a)$, node a approximates the column of a two-dimensional GHC. Thus, a finds m contact peers which are as close as possible to a on



(a) $n = 32$ (b) $n = 64$ (c) $n = 128$

Fig. 6.4: Example overlay graphs with random node IDs; $dim = 2$.

the 0-axis but span the interval $[0, 1)$ on the 1-axis:

$$\begin{aligned} Contacts_1(u) &= \{pred_0^{(l)}(a) | 0 \leq l \leq \left\lceil \frac{m}{2} \right\rceil\} \\ &\cup \{succ_0^{(l)}(a) | 0 \leq l \leq \left\lceil \frac{m}{2} \right\rceil\} \end{aligned} \tag{6.10}$$

Finally, we can now define the overall set of contacts of node a:

$$Contacts(a) = Contacts_0(a) \cup Contacts_1(a) \tag{6.11}$$

Figure 6.4 shows examples of such two-dimensional overlays. It can seen that by increasing the number of nodes n, the overlay follows the shape of a grid with horizontal and vertical links approximating the shape of a two-dimensional GHC. The goal is that each node should be able to reach any other node with a combination of one horizontal link and an additional vertical link, or the other way around: one vertical link and an additional horizontal link.

6.2.3 Routing Algorithm

The construction of the routing tables in Section 6.2.2 provides already a rough idea how routing in a two-dimensional-GHC-based overlay should occur. Now, we describe the routing algorithm precisely.

Each node a has its own view of the network with $Contacts(a)$. We define a function $a.find\_node(key)$ which can be executed at each node a with $key \in [0, 1)^2$ as follows:

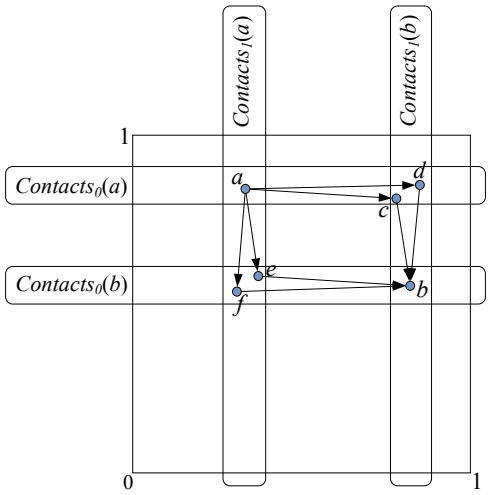


Fig. 6.5: Routing from node a to node b in the two-dimensional case.

$$a.find\_node(key) \;=$$

$$\begin{cases} \{b\} & if\ \exists\, b \in Contacts(a),\ b = key \\ \{pred_i(b, Contacts(a)), succ_i(b, Contacts(a)) \mid i \in \{0,1\}\} & otherwise \end{cases}$$

In other words, $a.find\_node(key)$ searches in $Contacts(a)$ for the closest two nodes (one predecessor and one successor) to key on each axis. The result can be a set of up to four nodes. For example, in Figure 6.5, $a.find\_node(b)$ returns the set of nodes $\{c, d, e, f\}$. This function is the basis for key-based routing in two-dimensional-GHC-based overlays as we will see below.

We describe the routing algorithm here in an iterative manner without loss of generality, i.e., the routing algorithm can be applied recursively with a little modification[1].

At each routing step, a locates the closest nodes to b on each axis from the nodes in $Contacts(a)$ or the nodes that it has learned about by asking other peers "on the way" towards b. The routing algorithm in provided in Algorithm 6.1. At the first iteration of the while loop (line 5 to 18), a locates nodes c, d, e and f (see Figure 6.5) by executing $find\_node(b)$

[1] Note however that iterative routing performs better from a resilience perspective as will be explained in Chapter 7

Algorithm 6.1 : Simplified low-diameter overlay routing algorithm.

Function: Routing
Input: Source node a, destination node b
Output: Node b or set of closest nodes to b at each axis

```
1: if a = b then
2:    Return b
3: end if
4: prev_contacts ← {a}
5: while True do
6:    new_contacts ← {}
7:    for u ∈ prev_contacts do
8:       new_contacts ← new_contacts ∪ u.find_node(b)
9:    end for
10:   if b ∈ new_contacts then
11:      Return {b}
12:   else if new_contacts\prev_contacts = ∅ then
13:      ("No new nodes encountered")
14:      Return prev_contacts
15:   else
16:      prev_contacts ← new_contacts\prev_contacts
17:   end if
18: end while
```

locally (lines 7 and 8) in Algorithm 6.1. In the 2nd iteration of the while loop, $prev\_nodes = \{c, d, e, f\}$ at line 7. Node a sends an RPC message $find\_node(b)$ to each node in $prev\_nodes$ ($u.find\_node(b)$ in line 8). Then each node in $prev\_nodes$ executes $find\_node(b)$ locally and sends the response back to a. Node a collects all responses in $new\_contacts$. At this point, the 'if' condition at line 10 yields True and the routing algorithm terminates successfully.

One drawback of this simplified routing algorithm is that the number of lookup messages grows exponentially at each routing step. However, using iterative routing, the lookup initiator a has full control of the number of messages sent at each iteration. Thus, a limits the number of lookup requests to four, by choosing the closest two nodes on each axis at each iteration.

6.3 Evaluation

In this section, we evaluate two-dimensional-GHC-based overlays, particularly the expected path length between arbitrary two nodes. The evaluation is based on a mathematical model which is validated by simulations.

6.3.1 Mathematical Model

Let $path\_length(a, b)$ the number of iterations of the while loop in Algorithm 6.1 required to reach the destination node b from source node a.

Theorem 6.1:

For two arbitrary nodes a and b,

$$\lim_{n\to\infty} P[path\_length(a, b) > 2] = e^{-2} \tag{6.12}$$

Proof Sketch:

From Figure 6.5 we can see that $path\_length(a, b) > 2$ if and only if

$$Contacts_0(a) \cap Contacts_1(b) = \varnothing \text{ and } Contacts_0(b) \cap Contacts_1(a) = \varnothing$$

As we mentioned above, a has determined m contact peers on the 0-axis, $Contacts_0(a)$, as the immediate neighbors to a on the 1 axis. Given, that u_0 and u_1 are independent for all $u \in V$, the selection of the set

$$\{u_0 \,|\, u \in Contacts_0(a)\}$$

among $\{u_0 \,|\, u \in V\}$ is equivalent to the selection of m out of n numbers randomly and without replacement. Thus, the probability distribution of the number of selected nodes $u \in Contacts_0(a)$ which fall into $Contacts_1(b)$ can be modeled by a hypergeometric distribution: We draw m balls ($Contacts_0(a)$) from n balls randomly without replacement. m of the n balls are white (those in $Contacts_1(b)$) and the rest is black. Thus, the probability of having k white balls at the end of the experiment is:

$$P[\,|\,Contacts_0(a) \cap Contacts_1(b)|\, = k\,] = \frac{\binom{m}{k} \cdot \binom{n-m}{m-k}}{\binom{n}{m}} \tag{6.13}$$

Let $k = 0$. Then,

$$\begin{aligned} P[Contacts_0(a) \cap Contacts_1(b) &= \varnothing] = \frac{\binom{n-m}{m}}{\binom{n}{m}} \\ &= \frac{(n-m)!}{m! \cdot (n-2m)!} \cdot \frac{m! \cdot (n-m)!}{n!} \\ &= \frac{(n-m)!}{(n-2m)!} \cdot \frac{(n-m)!}{n!} \\ &= \frac{n-m}{n} \cdot \frac{n-m-1}{n-1} \cdot \ldots \cdot \frac{n-2m+1}{n-m+1} \\ &= (1-\frac{m}{n}).(1-\frac{m}{n-1}).\ldots.(1-\frac{m}{n-m+1}) \\ &= \prod_{j=0}^{m-1}(1-\frac{m}{n-j}) \end{aligned} \tag{6.14}$$

Now, we determine an upper and lower bounds for

$$P[Contacts_0(a) \cap Contacts_1(b) = \varnothing].$$

$$\begin{aligned} P[Contacts_0(a) \cap Contacts_1(b) = \varnothing] &\leq \prod_{j=0}^{m-1}(1-\frac{m}{n}) \\ &\leq (1-\frac{m}{n})^m \\ &\leq (1-\frac{1}{m})^m \end{aligned} \tag{6.15}$$

It is well known that $(1-\frac{1}{m})^m$ converges to e^{-1} when $m \to \infty$.

On the other hand,

$$\begin{aligned} P[Contacts_0(a) \cap Contacts_1(b) = \varnothing] &\geq \prod_{j=0}^{m-1} (1 - \frac{m}{n-m}) \\ &= (1 - \frac{m}{n-m})^m \\ &\geq (1 - \frac{m}{m^2-m})^m \\ &\geq (1 - \frac{1}{m-1})^m \\ &\geq (1 - \frac{1}{m-1})^m \\ \geq (1 - \frac{1}{m-1})^{m-1} &\cdot (1 - \frac{1}{m-1}) \end{aligned} \tag{6.16}$$

The term on the right hand side of the last inequation in (6.16) converges to e^{-1} as well when $m \to \infty$. Thus,

$$\lim_{n\to\infty} P[Contacts_1(a) \cap Contacts_0(b) = \varnothing] = e^{-1} \tag{6.17}$$

For large values of n, the events $(Contacts_0(a) \cap Contacts_1(b) = \varnothing)$ and $(Contacts_1(a) \cap Contacts_0(b) = \varnothing)$ can be considered as independent. Thus,

$$\lim_{n\to\infty} P[p(a,b) > 2] = e^{-2} \tag{6.18}$$

∎

Tolerance Coefficient

According to the overlay construction above, $P[path\_length > 2]$ converges to $e^{-2} \approx 0.135$ which is still high depending on the application requirements. In order to reduce the probability that $P[path\_length > 2]$ to an arbitrarily small ϵ, a tolerance coefficient α can be introduced in the collection of contact nodes: for $i = 0, 1$ and $j = 0, 1$, $i \neq j$,

$$\begin{aligned} Contacts_i(u) &= \{pred_j^{(l)}(a) | 0 \leq l \leq \left\lceil \frac{\alpha \cdot m}{2} \right\rceil\} \\ &\cup \{succ_j^{(l)}(a) | 0 \leq l \leq \left\lceil \frac{\alpha \cdot m}{2} \right\rceil\} \end{aligned} \tag{6.19}$$

Theorem 6.2: *For tolerance coefficient α and two arbitrary nodes a and b*

$$\lim_{n\to\infty} P[path\_length(a,b) > 2] = e^{-2\alpha^2} \tag{6.20}$$

Thus, in order to keep the probability under ϵ, α must fulfill $\alpha \geq \sqrt{-\frac{log(\epsilon)}{2}}$.

Proof Sketch: The proof sketch is similar to the one of Theorem 6.1. In equation(6.14), we substitute the number of nodes on each row or column m by $\alpha \cdot m$. Thus,

$$P[Contacts_0(a) \cap Contacts_1(b) = \prod_{j=0}^{\alpha \cdot m-1} (1 - \frac{\alpha \cdot m}{n-j}) \tag{6.21}$$

Then, we determine upper and lower bounds for

$$\prod_{j=0}^{\alpha \cdot m-1} (1 - \frac{\alpha \cdot m}{n-j})$$

which lead to

$$\lim_{n \to \infty} P[Contacts_0(a) \cap Contacts_1(b) = \varnothing] = e^{-\alpha^2} \tag{6.22}$$

Thus,

$$\lim_{n \to \infty} P[path\_length(a,b) > 2] = (e^{-\alpha^2})^2 = e^{-2\alpha^2} \tag{6.23}$$

∎

Figure 6.6 shows how $\lim_{n \to \infty} P[path\_length(a,b) > 2]$ depends on α according to Theorem 6.2.

- For $\alpha = 1$, $P[path\_length(a,b) > 2]$ converges to $e^{-2} \approx 0.135$ as proven in Theorem 6.1.
- For $\alpha = 0$, routing tables are empty and routing does always fail leading to $P[path\_length(a,b) > 2] = 1$.
- For $\alpha = 2$, $P[path\_length(a,b) > 2]$ converges to $e^{-8} \approx 3.35 \cdot 10^{-4}$.

6.3.2 Simulations

We implemented the routing Algorithm 6.1. Coordinates of the nodes are chosen uniformly at random. First, we simulate a network with $n = 10^3$ nodes and tolerance coefficient $\alpha \in \{1, 2\}$. We compute the path length for n out of n^2 random source and destination $(a, b) \in V^2$. We repeat the experiment five times. Figure 6.7 shows the path length distribution.

In all cases, the routing algorithm terminates. The values of $P[path\_length(a,b) > 2]$ are inline with Theorem 6.2.

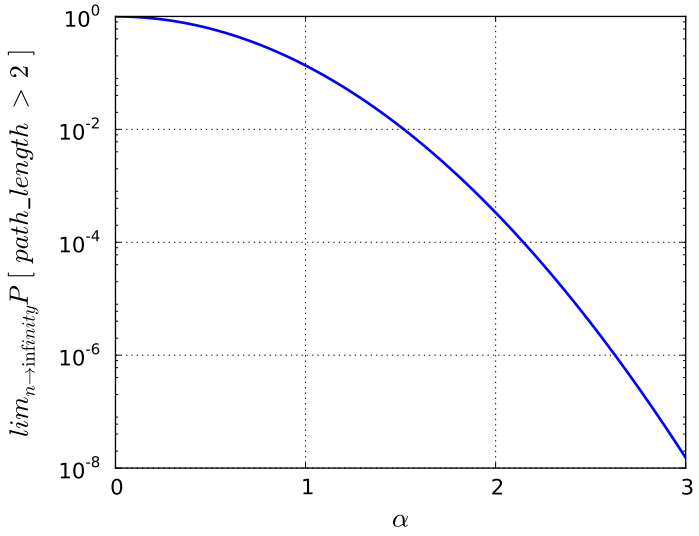


Fig. 6.6: The probability of routing in more than two hops depending on the tolerance coefficients α.

In the second simulation, we vary the number of nodes in the network n from 10^3 to $4 \cdot 10^3$ with additive increments of 10^3 and the tolerance coefficient α between 0 and 2 with additive increments of 0.1. For each value pair n and α, we perform 5 simulations runs, in each simulation run, n out n^2 pair of nodes are chosen randomly and the path length is measured. Figure 6.8 shows the Quantile-Quantile plots (Q-Q plots) which compare the simulation results with the model.

We can observer that by increasing n, the measured values of $P[path\_length(a,b) > 2]$ lie closer to the line $y = x$ which is due to the convergence behavior. The larger n is, the closer $P[path\_length(a,b) > 2]$ is to $e^{-2\alpha^2}$.

6.4 Supervised Low-Diameter Overlays

Finally, we present a optimization of two-dimensional-GHC-based overlays is inspired by the supervised node ID introduced in Chapter 5.

6.4.1 Approach

In Chapter 5, we introduced an approach for node ID assignment where node IDs follow the sequence a_i, $i \in \mathbb{N}$ which is defined by:

$$0 = 1 \prec \frac{1}{2} \prec \frac{1}{4} \prec \frac{3}{4} \prec \frac{1}{8} \prec \frac{5}{8} \prec \ldots \tag{6.24}$$

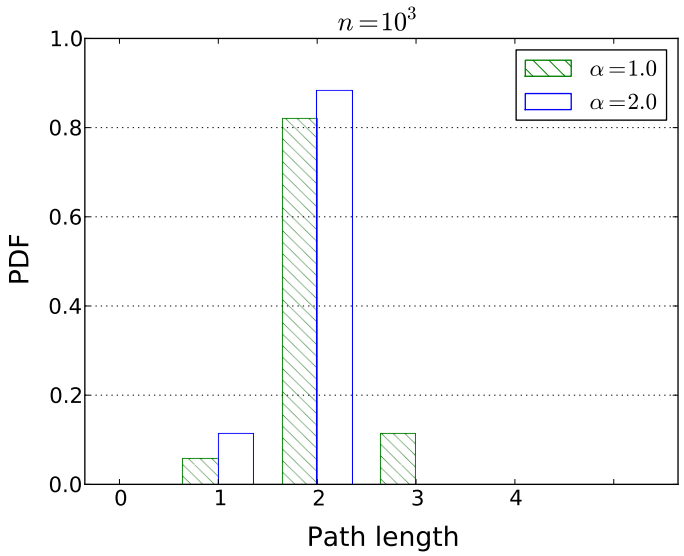


Fig. 6.7: Path length PDF in two-dimensional overlays

The sequence a_i can be extended to produce pairs of real numbers in $[0,1)^2$ using *pairing functions*, i.e., bijective function $\phi : \mathbb{N} \mapsto \mathbb{N}^2$. Figure 6.9 shows such a pairing function ϕ[2]. Algorithm 6.2 provides the algorithm which generates a sequence of pairs (i,j) following ϕ.

Algorithm 6.2 : Algorithm for reverse pairing function $\phi : \mathbb{N} \mapsto \mathbb{N}^2$.

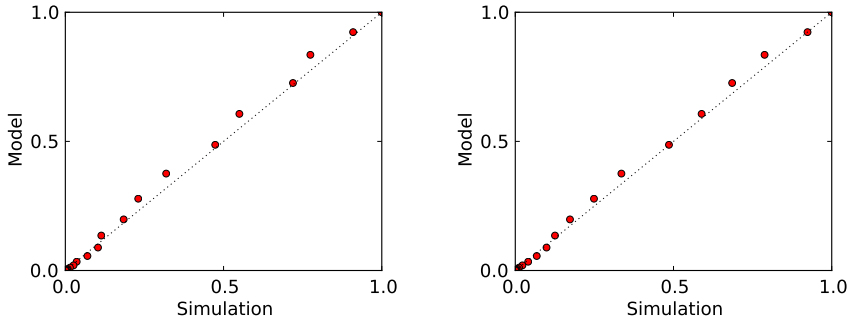
Function: Pairing Function ϕ

```
1: i ← 0
2: while True do
3:    for j ∈ {0, ..., i + 1} do
4:       yield (i, j)
5:    end for
6:    for j ∈ {1, ..., i + 1} do
7:       yield (i − j, i)
8:    end for
9:    i ← i + 1
10: end while
```

Let $\psi : \mathbb{N} \mapsto [0,1)^2$ be defined such as

$$\psi(k) = (a_i, a_j) \qquad where \quad \phi(k) = (i,j) \tag{6.25}$$

[2] Note that we do not use the Cantor pairing function [143], since the numbers generated by the Cantor pairing function are not efficient for our purposes, namely routing within two hops.



(a) $n = 1000$ (b) $n = 2000$

(c) $n = 3000$ (d) $n = 4000$

Fig. 6.8: Quantile-Quantile plots comparing the simulation results for $P[path\_length(a, b) > 2]$ with the model in Theorem 6.2.

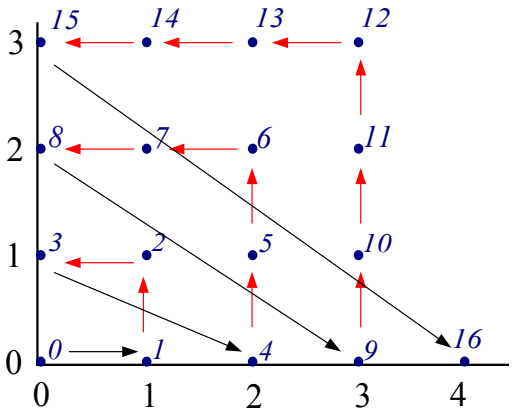


Fig. 6.9: Pairing function $\phi : \mathbb{N} \mapsto \mathbb{N}^2$.

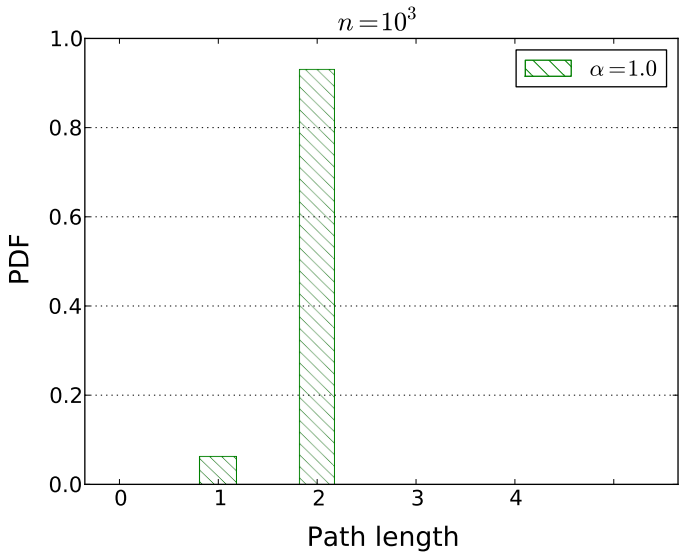


Fig. 6.10: Path length PDF in two-dimensional overlays with near-optimal node IDs

ψ generates near-optimal coordinates for nodes in the two-dimensional address space $[0, 1)^2$.

6.4.2 Evaluation

Figure 6.10 shows the simulation results with supervised near-optimal node IDs, again with $n = 10^3$. $P[path\_length(a, b) > 2]$ is reduced to zero.

6.5 Related Work

Most well know DHT algorithms perform routing within $O(log\, n)$ hops, e.g., Chord [144], Pastry [123] and Kademlia [123]. Comparative analysis of these DHTs in terms of degree vs. diameter optimality is provided in [49, 81]. It is typical for large server farms to either have some nodes with a special role for the coordination or use a broadcast mechanism, thus building a structure which can be considered as one hop DHT. Explicit algorithms for one hop DHTs have been suggested in [78, 80, 90]. In [145], a traffic analysis reveals that one hop DHTs are appropriate for overlays with up to few thousand nodes. However, the signaling overhead for the overlay maintenance becomes too high for overlays with up to a million nodes. The most relevant work on two-hop DHTs has been Kelips [51] and two-hop calot DHTs [145]. Kelips maintains $O(\sqrt{n})$ routing tables and achieves routing within $O(1)$. However, for this purpose a system parameter $k = \lceil \sqrt{n} \rceil$ needs to be determined beforehand. In contrast to Kelips, GHC-based overlays

defined in this chapter allow for the network to grow dynamically and nodes to adapt to the network size by estimating the number of nodes in the network n and deducing the number of contacts they need in their routing tables $2\alpha m$.

The approach of two-hop calot DHTs [145] is the closest to ours. The authors propose using two overlapping rings. Each node has two different node IDs on each ring. This is equivalent to the two coordinates of each node in GHC-based overlays. However, at each routing step, a node determines which one of its virtual nodes is closer to the destination on the respective ring. In GHC-based overlays, this would be equivalent to choosing the axis where the destination is closer. In contrast to two-hops calots, routing in GHC-based overlays benefits from different paths following different permutations of the axes. This allows for parallel lookups. Thus, GHC-based overlays are more resistant against stale routing table entries.

Finally, none of the previous work on two-hop DHTs, including Kelips and two-hops calots, have investigated the benefits of supervised near-optimal node IDs as discussed in this chapter.

6.6 Conclusions

In this chapter, we have questioned the myth that DHTs routing is always in $O(log\, n)$. We presented a DHT algorithm based on two-dimensional Generalized Hypercubes (GHCs). It provides lookup in two hops with probability $(1-e^{-2})$ for random node IDs. The routing table size is $2\cdot\sqrt{n}$. Furthermore, the probability that routing within two hops succeeds can be increased to $(1-\epsilon)$ for an arbitrarily small ϵ by increasing the routing table size by the factor of a tolerance coefficient $\alpha = \sqrt{-\frac{log(\epsilon)}{2}}$. The mathematical model is validated via simulations.

In Chapter 5, we introduced near-optimal node IDs in the one-dimensional ID space $[0, 1)$ for improving load balancing. In this chapter, we extended the supervised node IDs approach to generate near-optimal node IDs in $[0, 1)^2$ and improve the routing performance of two-dimensional-GHC-based overlays to guarantee that routing succeeds within two hops.

Given the current trend with mega-scale server farms of up to million servers [20, 32, 50, 95] which are potentially distributed all over the world, two-hop DHTs seem to be a compromise between one hop DHTs which would incur high overlay maintenance overhead, and $O(log\, n)$ DHTs which do not fulfill the latency requirements for today's applications such as search engines [20] and online shopping [32]. GHC-based overlays allow for parallel lookup, which allows for proximity-based routing and makes the routing resilient against stale routing table entries.

Part III

SECURITY AND PRIVACY

7. COSIP SECURITY

In Chapter 4, we introduced CoSIP, a hybrid architecture for SIP signaling with a server and a P2P network in parallel. Since routing and storage in the P2P network are performed by peers which are not necessarily trustworthy, we need to investigate the security implications of such an architecture. Particularly, we need to address the following questions:

- Does the P2P network introduce new attack vectors?
- Can attackers invalidate the assumptions on the reliability benefits of the P2P network?
- To what extend can CoSIP as a supervised P2P network approach address the security issues inherent in P2P networks and what other security mechanisms are still necessary to provide a resilient SIP signaling solution?

The rest of this chapter is organized as follows. Section 7.1 provides security requirements for a resilient SIP signaling solution. Section 7.2 provides a formal threat model. Section 7.3 describes security mechanisms, notably secure node ID assignment and resilient overlay routing. Section 7.5 provides an overview of related work and Section 7.6 concludes this chapter. In Appendix C which is closely related to this chapter, an extensive threat analysis of P2P networks, notably in the context of CoSIP, is provided to validate to what extent the proposed security mechanisms do adequately address the security issues.

7.1 Security Requirements

We deduce a list of security requirements directly from the security CIA model introduced in Chapter 2. Thus, a "secure" solution for SIP signaling needs to fulfill the following security requirements:

- *Confidentiality*: absence of unauthorized disclosure of information.
- *Integrity*: absence of improper system alterations.
- *Availability*: readiness for correct service.

This list of security requirements is concise though fulfilling it is not trivial for the following reasons:

- Message confidentiality and integrity of the SIP signaling message and VoIP streams can be achieved upon successful mutual authentication and key establishment between SIP endpoints (Callers and Callee). However, in current SIP signaling solutions SIP endpoints authenticate themselves only towards SIP proxies and registrars. A solution which allows for P2P signaling must allow for mutual end-to-end authentication between SIP endpoints based on their SIP identities.

- Integrity, i.e., absence of improper system alterations in the SIP signaling, is not only about message integrity. It requires also efficient mitigation of SPIT phone calls. Note that a SIP message can be correct while the call is actually illegitimate. Thus, message integrity is not sufficient to mitigate SPIT.

  In server-based SIP, the number of `INVITE` messages from a UA can be compared with a threshold value to detect if the UA is trying to initiate too many phone calls. Other more sophisticated SPIT detection and prevention can be deployed as well [94]. If phone calls are initiated at a P2P basis, it is very difficult to detect such a malicious behavior.

- Confidentiality is not only about encrypting the SIP signaling messages and VoIP streams. It must be assured that no private information about the user is revealed, e.g., location and social interaction in terms of with whom she is communicating. Thus, confidentiality includes also *privacy*.

  Using server-based signaling solutions, only the entities from the SIP infrastructure (proxies and registrars) involved in the signaling dispose of information about the users such as their location and their social interaction. The use of a P2P network for SIP signaling introduces new attack vectors if user location and information about her social interaction are available for arbitrary peers. Given the complexity of this issue, a separate chapter is dedicated to privacy in CoSIP and P2PSIP (Chapter 8) where a solution is provided in details and extensively evaluated.

- The third security requirement is availability. Availability of the SIP signaling using the P2P overlay requires the integrity and availability of:

  - Overlay signaling, i.e., messages for DHT storage, DHT lookup and overlay maintenance,
  - DHT content.

In order to provide a systematic approach to address these security requirements, a threat model is required. This is the subject of the next section.

Before we move forward with the threat analysis, we would like to mention for completeness that there exist potentially more security requirements in a SIP network which are not addressed in this chapter, e.g., accounting and lawful interception.

7.2 Threat Model

A common threat model for the security analysis of network protocols is the *Dolev-Yao* threat model [36]. We first introduce the Dolev-Yao threat model. Then, we introduce a slightly different variant which we will use for our threat analysis.

According to the Dolev-Yao threat model, an attacker Malice

- Can obtain any message passing through the network
- Is a legitimate user of the network, and thus in particular can initiate a conversation with any other user
- Will have the opportunity to become a receiver to any other entity in the network
- Can send messages to any entity by impersonating any other entity in the network. In the context of CoSIP, this means messages either at the overlay layer or the application layer (SIP).

One can think of any message sent to the network as being sent to Malice for her disposal; and any message received from the network as being received from Malice after her disposal.

This assumption is particularly useful in the context of P2P networks since overlay routing and storage are performed by the network participants. An attacker can be another peer in the overlay. Thus, we need to cope with attacks where peers along the overlay routing path may tamper with messages, inject or discard messages. Discarding is notably critical since we need to consider the free-riding problem where peers use the resources in the P2P network but do not contribute any resources. Thus, they can just discard any message they are supposed to forward. As for storage, one may think of a stored data item in the DHT as a message that is waiting for delivery. Thus, peers storing the data may tamper, inject or discard stored data.

Attacker's Scope Limitations

Note that the Dolev-Yao threat model does particularly not deal with attackers which have access to the memory of other network participants, e.g.,

due to a security flaw in the operating system. These attacks could be used to acquire access, e.g., to encryption keys which are supposed to be secret.

In the context of SIP and P2P networks, this restriction means also that attacks that target security flaws in the implementation of SIP UAs or the P2P protocol are out of scope in our threat analysis.

Weakening The Dolev-Yao Threat Model

Albeit an attacker does not have access to the memory of other network participants, the Dolev-Yao threat model attacker is still very strong. Notably, if an attacker controls all communication between two entities A and B, it is not possible to mitigate threats where the attacker just discards all communication. Such an attack would result into the complete loss of SIP signaling availability on top of the overlay. Instead of using an omnipresent attacker, we assume that an arbitrary node in the P2P network involved in forwarding a message from A to B or storing a data item m is malicious with probability p.

7.3 Mechanisms

In this section, we describe mechanisms for addressing the security requirements mentioned in Section 7.1 and according to the threat model introduced in Section 7.2.

7.3.1 Secure Node ID Assignment

The first fundamental tool required is *secure node ID assignment*. A malicious node may generate as many fake node identities in the overlay as possible to increase the probability that she will be responsible for forwarding a message or storing a data item. This malicious behavior has been denoted by *Sybil attack* [1]. Already in the original paper on Sybil attacks by Douceur [37], it was observed that "without a logically centralized authority, Sybil attacks are always possible except under extreme and unrealistic assumptions of resource parity and coordination among entities."

Therefore, a central authority is required in the network to provide verifiable IDs at the overlay layer. We use the CoSIP server as a central authority. It generates a node ID for each node upon successful user registration and embeds the node ID within an X.509 certificate. The CoSIP server needs to keep track of which user has already received which node IDs, in order to limit the number of legitimate node IDs per user. Peers use the X.509

[1] John R. Douceur [37] was the first who introduced the name "Sybil attack" after a novel about a woman whose name was Sybil [125]. She suffered under multiple personalities. Her mind broke apart and compartmentalized her personality to fifteen other "selves".

certificates to mutually authenticate each other at the overlay layer, prove the correctness of their respective node IDs, and establish keys for integrity-protection of the overlay signaling.

Moreover, in Section 7.1, we mentioned that a solution which allows for end-to-end SIP signaling must allow for mutual end-to-end authentication between SIP endpoints. Thus, the CoSIP server provides UAs with verifiable identities at the application layer as well, in form of X.509 certificate including the SIP identity. SIP UAs use these certificates to mutually authenticate each other and establish keys for encryption and integrity-protection of the SIP signaling and subsequent multimedia streams in an end-to-end manner. Moreover, the contact data of a user Alice stored in a DHT can be cryptographically signed by Alice.

Authentication Protocol

The two verifiable identities required by a UA are provided by the CoSIP server upon successful user registration. An authentication protocol which runs between a CoSIP endpoint and the CoSIP server provides this functionality. The authentication protocol assumes that the CoSIP Server, or Authentication Server AS, shares a long-term pre-shared key $PSK_{AS,A}$ with UA A. The pre-shared key can be a user password, or a high entropy key stored in the (U)SIM card of the user's smart phone. After successful user authentication, the AS generates the node ID a. It provides the UA with a certificate which binds the node ID a to the public key K_a and a certificate which binds the user's public key $+K_A$ to her SIP URI A.

Figure 7.1 outlines the CoSIP setup with the AS and the UAs communicating to each other. In fact, this figure shows the difference between the CoSIP authentication protocol and Identity Management (IdM) protocols [101]. In IdM protocols, clients acquire credentials from an Identity server to authenticate themselves towards other servers. In the CoSIP case, SIP UAs acquire credentials from the AS (the overlay supervisor) to authenticate themselves to each other. In the first step of the protocol, client A and AS perform a TLS handshake where the AS is authenticated using a certificate. The server certificate needs to be signed by a common trust anchor, e.g., a root CA which is trusted by both A and AS. Symmetric keys are generated as as side effect of the TLS handshake on both sides for encryption and integrity protection of the subsequent messages 1 to 4 in message flow (7.1) below.

Let r be a random number freshly generated by AS at each protocol run. Further, let $Cert_{AS}(e, +K_e)$ denote a digital certificate for e issued by AS. We abstract from details, e.g., certificate validity intervals. h is a cryptographic hash function, and '|' the concatenation operator. A successful

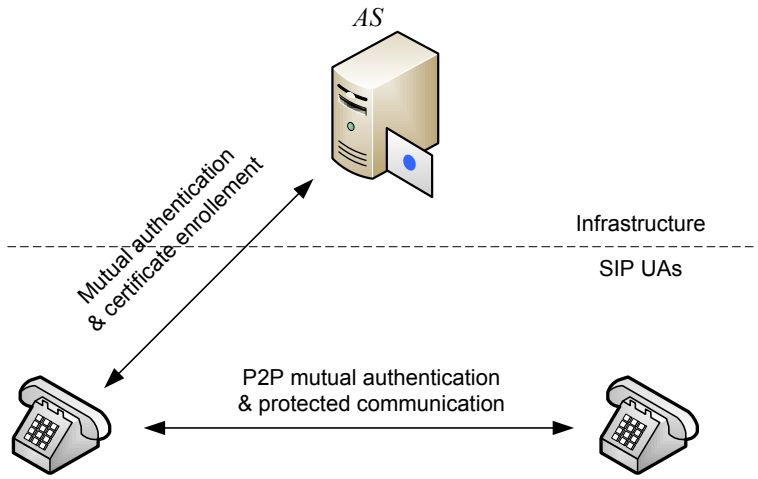


Fig. 7.1: P2SIP Identity Management

authentication protocol run then looks as follows:

$$\begin{aligned} 1: \quad & A \rightarrow AS \quad : \quad Identity\_req \\ 2: \quad & A \leftarrow AS \quad : \quad r \\ 3: \quad & A \rightarrow AS \quad : \quad h(\, A \,|\, PSK_{A,AS} \,|\, r \,), +K_A, +K_a \\ 4: \quad & A \leftarrow AS \quad : \quad Cert_{AS}(A, +K_A), Cert_{AS}(A, +K_A) \end{aligned} \tag{7.1}$$

A successful run of the protocol enables A to authenticate herself towards the server, and then acquire the two required verifiable identities.

We implemented the authentication using SIP as "transport protocol". The SIP user registration procedure is enhanced with the certificate enrollment. The implementation details are provided in Appendix B.

7.3.2 Resilient Routing

Second, we address the problem of maliciously discarding overlay lookup messages.

Parallel Lookups

Figure 7.2 shows how a lookup message is forwarded via one hop in parallel from A to B. Let k be the number of parallel nodes between A and B and p be the probability that an arbitrary peer is malicious. Thus,

$$P[routingsuccess] = 1 - p^k \tag{7.2}$$

For example, if A sends $k = 3$ parallel requests to B, and $p = 0.2$, this results into a routing success probability of 0.992. Since this success

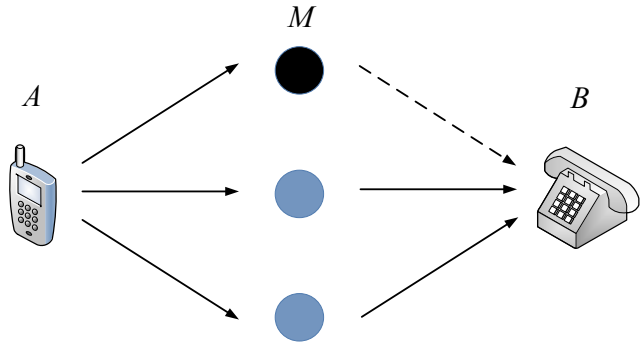


Fig. 7.2: Parallel routing in DHTs with an attacker on one of the paths.

probability might still be not sufficient, A may probe different paths in a wider scope if routing fails. This is the case, e.g., for Kademlia [86] where the number of parallel lookup requests is increased to the size of the k-buckets, e.g., $k = 8$, or $k = 20$.

Iterative vs. Recursive Routing

Using *recursive overlay* routing, a message is forwarded from a source A to a destination D hop by hop, e.g., via nodes B and C as follows:

$$A \rightarrow B \rightarrow C \rightarrow D$$

The response is either sent directly to A or hop-by-hop backwards taking the same path:

$$D \rightarrow C \rightarrow B \rightarrow A$$

In the latter case, the routing is *symmetric recursive routing*. Using *iterative routing*, the first hop on the path B does not forward the lookup message to C. Instead, it responds to A providing the contact data (IP and port) of C. Then, A contacts C directly and so on.

Figure 7.3 shows how a message is forwarded from A to B recursively.

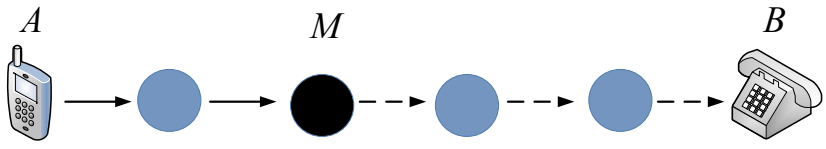


Fig. 7.3: Recursive routing in DHTs with an attacker on the path.

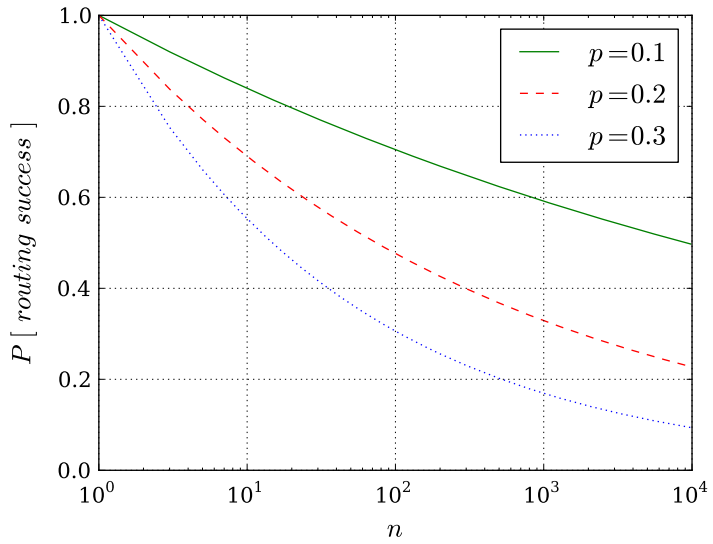


Fig. 7.4: Routing success probability using recursive routing with different attacker ratios p.

Let k be the number of hops between A and B. Thus,

$$P[routingsuccess] = (1-p)^k \tag{7.3}$$

Taking the Chord DHT algorithm [144] as an example, the average path length between two nodes has been proven to be $^{log_2(n)}/_2$ [81]. Thus, let's assume $k = {}^{log_2(n)}/_2$. Figure 7.4 shows the routing success probability for different values of p.

As we observe in Figure 7.4, the routing success probability drops quickly by increasing n. A routing availability of "5 nines" is far from being reached.

A may try another parallel path as described above. However, there are still two problems faced with recursive routing:

Path diversity: Since A has only contact to the first hop on the path, she can not identify which peer is to blame if a message gets lost. She does not even know the other peers on the path. Thus, she can not guarantee that parallel lookups do not converge towards an attacker on the path as shown in Figure 7.5. In other words, A is not able to identity multiple node-disjoint paths to guarantee that parallel lookups would bring any improvement.

Note that the same problem is encountered in case peers are not malicious but just go offline without notifying their neighbor peers.

Lookup latency: Since A can only roughly estimate the path length and does not have any information about the round trip time between every two

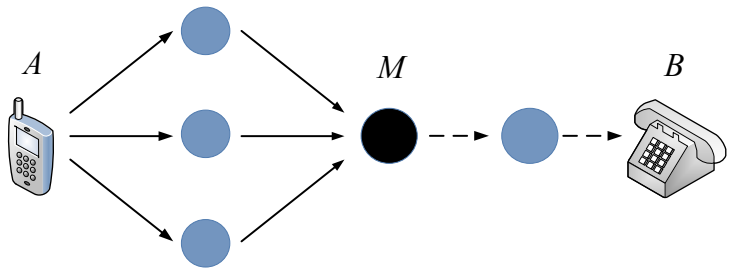


Fig. 7.5: Parallel recursive routing in DHTs with an attacker on the path.

hops on the path, it is very difficult to estimate the overall lookup latency and to setup an appropriate timeout interval to re-initiate the lookup. This leads to a waste of time.

To conclude, by simply discarding all signaling messages, an attacker can cause a serious harm to the availability of the overlay routing. The consequences are higher lookup latency and a significant degradation of lookup availability. The solution is twofold:

- Parallel lookups to increase the probability to successfully route beyond faulty nodes.
- Iterative routing to better control the lookup routing progress. This reduces the required lookup request timeouts, and allows for detecting faulty nodes more efficiently and routing beyond them.

7.3.3 Data Replication

For the purpose of completeness, we mention data replication as a mechanism for addressing the threat model from Section 7.2 since attackers may simply discard data items instead of storing them.

As discussed in Chapter 4, data replication is inherent in P2P networks to provide high reliability. If A publishes her contact data at k nodes and $j \leq k$ nodes among the replica nodes are malicious, this reduces the availability of A's contact data in the DHT. However, given that secure node ID assignment provides protection against Sybil attacks, we assume that a malicious peer can not control more than one peer in the overlay. Thus, the incentives for a malicious peer to discard data are minimal for the following reasons:

- The probability that discarding the data would successfully lead to the unavailability of the contact data of A is very low, since there are sufficiently other replica nodes in the DHT that would be able to deliver it.

- Storing the contact data of A does not require a considerable amount of resources.

7.4 Evaluation

An extensive threat analysis of P2P networks, notably in the context of CoSIP is provided in Appendix C.

7.5 Related Work

Security issues in P2P networks have been initially described in [136]. The P2P research community has then spent a considerable effort on addressing these security issues in a distributed manner [11, 31, 33, 34, 83, 150]. However, these solutions can not provide resilience guarantees. One of the main problems is that they do not provide protection against Sybil attacks. For example, [11] provides a mechanism to route DHT lookups beyond malicious nodes and show that if 20% of the nodes in the network are malicious, the lookup success rate is at 99.0%. This is obviously far from being sufficient for reliability telephony. First a lookup availability of 99.0% is not sufficient. Second, an attacker with sufficient resources can easily launch a Sybil attack and emulate more than 20% of the nodes in the network.

Further research effort exclude security issues in their work by simply proposing the use of a PKI [22, 124], but do not specify how such a PKI can solve the security problems. Seedorf [128] discusses the security issues inherent in P2PSIP. In [129] he proposes self-certifying SIP-URIs. A SIP-URI is generated based on a cryptographic hash value of the user's public key. This can be useful to protect the integrity of the DHT content. However, it does not provide any protection against Sybil attacks or attacks on the routing. Moreover, cryptographic SIP-URIs are annoying from usability perspective. In [130], Seedorf investigates the challenges of lawful interception (LI) in P2PSIP networks, and potential solutions. In [12], the authors investigate a game theoretical approach for the security threats of P2PSIP such as SPIT and attacks on overlay routing.

The IETF P2PSIP base protocol RELOAD specifies the enrollment of certificates from an enrollment server. However, a certificate is used for both user and node authentication. It is well known [41] that different keys should be used for different purposes. Thus, in contrast to RELOAD we use different certificates for the two different verifiable identities.

While our analysis of the negative impact of recursive routing on the routing resilience is comprehensible, recursive routing is still commonly believed to be an option in P2P design. For example, the RELOAD draft specifies the use of symmetric recursive routing. The main motivation behind symmetric recursive routing is that the request initiator does not need

to perform the required NAT traversal procedure [119] to guarantee IP connectivity with each of the nodes on the path during the iterative lookup. This approach may be useful to save traffic and lookup latency if the network is stable and the probability that there are malicious peers on the path is very low. However, unless the P2P network would be a network of servers (in which case NAT traversal would not be a problem) where all nodes are trustworthy, this assumption seems to be unrealistic to us.

The security solutions for CoSIP presented in this chapter have been designed with a strong focus of the resilience and the reliability of the service provided by the overlay. Thus, although there has been a good deal of work on the topic of the security of P2P networks, we consider the security analysis and mechanisms discussed in this chapter as distinctive from other work in these areas and appropriate in the context of resilient IP telephony.

7.6 Conclusions

CoSIP uses a P2P network as a backup for session establishment. Thus, it is critical to prevent attacks which aim at invalidating the reliability benefits of the additional P2P network. These attacks may target the overlay routing or the DHT content. Being a supervised P2P network approach, CoSIP addresses the security issues in P2P partially by providing secure node ID assignment. This addresses the issues of Sybil attacks, chosen-location attacks and eclipse attacks. Moreover, verifiable SIP URIs allow for P2P mutual authentication and establishment of security context at the application layer; and integrity-protection of DHT content by digital signatures.

However, attacks on the P2P network can not simply be resolved by adding some cryptography. Notably, even if all overlay messages are integrity-protected, a malicious node responsible for forwarding a message may forward it to the wrong peer, or may simply discard it. A malicious node responsible for storing a data item may simply discard it or refuse to deliver it. Therefore, the resilience of P2P networks requires additional mechanisms such as redundancy in routing (parallel lookups) and storage (replica nodes). Moreover, iterative routing allows for the limitation of the impact of attacks on the routing and prevent the amplification of flooding attacks.

Unfortunately, the P2P network does introduce new attacks as well. Two issues are difficult to solve, namely SPIT and attacks on privacy. SPIT attacks are easier in a P2P mode. One way to counter this problem is to accept phone calls only from known UAs or to exploit knowledge from social networks to evaluate a SPIT score for incoming calls. However, anti-SPIT mechanisms at the infrastructure are expected to perform better than on a P2P basis. This confirms the fundamental design decision of CoSIP to use the P2P network only in parallel to the infrastructure and not as the single solution.

Attacks on privacy result from the fact that user location is stored in

the DHT and available for all network participants. Moreover, session establishment involves arbitrary peers which are able to deduce that some users have a social interaction. A solution for the privacy issues is provided and extensively evaluated in Chapter 8.

Finally, we would like to emphasize the strong interrelationship between reliability and security. As observed, an attacker can invalidate the reliability estimation of the P2P network that we computed in the reliability analysis in Chapter 4. For example, storing an object at k replica nodes provides redundancy, but becomes useless if all k replica nodes are controlled by a single malicious node launching a Sybil attack. The mechanisms proposed for addressing the security issues are typical mechanisms for enhancing the reliability of a system or a network. For example, data replication or multipath (parallel) routing [72]. This observation confirms the validity of our overall approach in this thesis to consider both disciplines (security and reliability) in joint efforts and the resilience of a system in a holistic approach.

8. PRIVACY-PRESERVING P2PSIP

In CoSIP (Chapter 4) as well as P2PSIP [98] the location of a SIP UAs (IP address and port number) is published in a DHT. This data is stored at other peers with peer identifiers (IDs) uncorrelated to the SIP UA. These peers, called replica nodes, reply to queries from any other peer looking for the UA. This makes the UA available for incoming VoIP phone calls and chat messages, notably in case of server failures in CoSIP. However, the SIP UA has no control over knowing which peers have asked for its current location. Curious and malicious peers can perform a lookup for the SIP URI of the UA regularly. The IP addresses of the UA could then be mapped to geographic locations [44, 85]. Using this information, attackers could build location profiles of a user. Even worse, attackers could "crawl" the P2P network and harvest location profiles of all participants. This severe privacy issue is valid for CoSIP as well as P2PSIP. Though, it has been left out-of-scope in the IETF P2PSIP working group (WG) [98]. On the other hand, location privacy had been thought of early in the GSM standardization process. Thus, it seems to be necessary to consider this privacy issue in CoSIP and P2PSIP networks as well.

Another privacy threat in CoSIP and P2PSIP is that during session establishment replica peers and potentially other nodes involved in the DHT lookup can observe that communication is established between two SIP UAs and deduce knowledge about the social interaction of the two users.

In this chapter, we tackle *location privacy* and *social interaction privacy* in CoSIP as well as P2PSIP by developing a new protocol: *Privacy-Preserving P2PSIP* (*Pr*^{2}*-P2PSIP*). The rest of this chapter is organized as follows. In Section 8.1, we present the design of Pr^2-P2PSIP. Section 8.2 provides an evaluation of Pr^2-P2PSIP in terms of threat analyses as well as an analysis of the overhead of adding privacy to P2PSIP networks in terms of cryptographic overhead, signaling latency and reliability costs. Section 8.3 provides an overview of related work and Section 8.4 concludes our findings in this chapter.

8.1 Design of Pr^2-P2PSIP

In this section, we introduce Pr^2-P2PSIP.

Tab. 8.1: Notation

$+K_e$	Public key of an entity e
$-K_e$	Private key of an entity e
$K_{a,b}$	Shared secret key between entities a and b
$\{m\}_{K_{a,b}}$	Message m encrypted and integrity-protected with the symmetric key $K_{a,b}$ (See Section 8.1.4).
$\{m\}_{+K_e}$	Message m encrypted with the public key of entity e.
$Cert_{AS}(e, +K_e)$	Digital certificate for e issued by AS. We abstract from details, e.g., certificate validity intervals .
$l(e,t)$	Location (IP address and port number) of the entity e at a certain point of time t
$L(A,t)$	Data stored in P2P network required to reach UA A at a certain point of time t

8.1.1 Model and Notation

First, we introduce the model and notation used in the rest of the chapter.

8.1.1.1 SIP UAs and Public Identities

The SIP UAs provide the means for users to perform their social interactions. They send chat messages and initiate phone conversations on behalf of the users. Let $\mathcal{N}$ be the set of UAs in a P2PSIP network and $n = |\mathcal{N}|$ the number of UAs. In this chapter, we use capital letters, e.g., A, B or $A_i, i \in \{1, 2, ...n\}$ to denote interchangeably (unless otherwise explicitly mentioned) a user name, her SIP UA, or her SIP URI. Table 8.1 provides additional notation used throughout this chapter.

8.1.1.2 Authentication Server

Pr^2-P2PSIP functions with a central authority, an *Authentication Server* AS. This can be the CoSIP server in CoSIP or the certificate enrollment server in P2PSIP. As described for the CoSIP case in Chapter 7, the AS provides the UA with a certificate $Cert_{AS}(A, +K_A)$ which binds the SIP identity A to the public key $+K_A$. Moreover, AS provides verifiable identities at the overlay layer. However, the subtle difference to the authentication protocol in non-privacy-preserving CoSIP networks described in Section 7.3.1

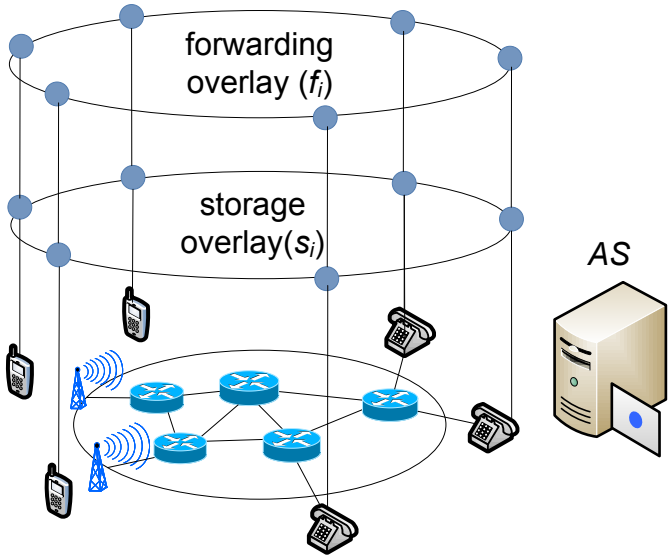


Fig. 8.1: Architecture of Pr^2-P2PSIP

is that the UA acquires two different identities at the overlay layer in Pr^2-P2PSIP, since Pr^2-P2PSIP functions with two different overlays as explained in the next Section 8.1.1.3.

8.1.1.3 Storage and Forwarding Overlays

In addition to its public identity, a UA A_i has two *pseudonyms* f_i and s_i which it uses for participating in two different overlays as sketched in Figure 8.1. $s_i, i = 1, \ldots, n$ is the *storage* overlay. $f_i, i = 1, \ldots, n$ is the *forwarding* overlay.

Storage

Storage is the common service that DHTs provide. The DHT stores information required to contact other UAs for sending them application layer signaling messages. However, the information stored in the Pr^2-P2PSIP DHT differs from P2PSIP. Specifically, it does not reveal the actual location of UAs. As noted in Table 8.1:

- $l(e,t)$ is the location (IP address and port number) of the entity e at a certain point of time t,
- $L(A,t)$ is the data stored in P2P network required to reach UA A at a certain point of time t.

Thus, in contrast to non-privacy preserving P2PSIP:

$$L(A,t) \neq l(A,t)$$

The actual information $L(A,t)$ stored in the DHT to reach A is explained below in Sections 8.1.2.2 and8.1.3.2.

Forwarding

Forwarding is an additional function that peers need to perform in Pr^2-P2PSIP. It differs from typical forwarding in DHT algorithms with recursive routing, e.g., Chord [144] or [123], given that these DHT algorithms were not designed with privacy in mind. Message forwarding in Pr^2-P2PSIP is explained in 8.1.2.1.

Overlay Algorithm

As for the non-privacy preserving CoSIP described in Chapter 4, we use Kademlia [86] as DHT overlay algorithm. However, Pr^2-P2PSIP could be used with other DHTs. We do not claim that the choice of the overlay algorithm is orthogonal to the impact of Pr^2-P2PSIP on user privacy. Thus, this design decision requires further investigation in future work. For this chapter, we use the Kademlia RPCs FIND_NODE, FIND_VALUE, PING and STORE in the storage overlay. Since the forwarding overlay is used only for finding other peers (i.e., no data stored in the DHT, see Section 8.1.2.1 for details), the forwarding overlay makes use only of the FIND_NODE and PING RPCs.

Pseudonyms in the Storage and Forwarding Overlays

The pseudonyms f_i and s_i are temporal identities which are unlinkable to the UA's public identity A_i (we use non-capital letters to denote pseudonyms). Pseudonyms f_i and s_i belong to an identifier space $\mathcal{K}$, e.g. $\mathcal{K} = \{0, \ldots, 2^{160} - 1\}$. Each pseudonym is linked to a public key as well: $(f_i, +K_{f_i})$, $(s_i, +K_{s_i})$. As such, a UA uses different public/private key pairs for different purposes.

By "UA A_i", we mean the UA with public identity A_i while "UA f_i" or "UA s_i" is the UA with pseudonym f_i or s_i respectively.

8.1.1.4 Threat Model

Given a UA $A \in \mathcal{N}$, we assume that an attacker M wants to collect as much information as possible about A, in particular:

1. its current locator $l(A,t)$
2. its location profile: a history of $l(A,t)$
3. a social interaction profile: a history of social interactions $A \rightarrow B$ or $B \rightarrow A$ for any $B \in \mathcal{N}$.

We consider the following attackers in Pr^2-P2PSIP:

1. a single malicious UA participating in the Pr²-P2PSIP network: $M \in N$. Since we can exclude Sybil attacks by verifiable identities (See Chapter 7), we assume that an attacker can not control more than one UA. Thus, we assume every UA operates on its own. Different malicious UAs do not exchange information for the sake of breaking other users' privacy. Thus, each UA can observe only the messages it sends and it receives. Additionally, if it forwards a message from one peer to another, it can decrypt only the messages (or message parts) for which it has the appropriate key.

2. a *partial* observer in the network underlay observing that communication is taking place between different IP addresses. The attacker may be able to observe some traffic and deduce some conclusions about the location or social interaction of some UAs.

8.1.2 Protocol Overview

In this section we describe how Pr²-P2PSIP handles data storage and message forwarding. Storage and forwarding in the Pr²-P2PSIP network differ from a "regular" P2PSIP network, because UAs seek to keep their location and social interaction private.

8.1.2.1 Message Forwarding

An application layer message (e.g., SIP MESSAGE for IM or SIP INVITE for establishing a phone call) from a UA A to a UA B is sent via intermediate forwarding peers using so-called *onion routing* [45]. In onion routing, the sender of a message m chooses intermediate forwarding peers which route the message to B on behalf of A. A orders these peers in series and encrypts m several times recursively. One layer of encryption is removed at each of the forwarding peers, so that the final peer in the tunnel has the original unencrypted message.

In Pr²-P2PSIP, peers establish *inbound tunnels* and *outbound tunnels* (see Figure 8.2). The choice of tunnel length has an impact on privacy which are discussed in detail in the evaluation Section 8.2. For illustration purposes, we consider a tunnel length of three hops throughout this design Section 8.1.

A UA A uses its pseudonym ($f_{O_0} = f_{I_0}$ in Figure 8.2) to communicate with the first hop of each tunnel. For outbound tunnels, A (sending application layer messages) generates symmetric keys for protected communication (i.e. encrypted and integrity protected) with each of the outbound forwarding peers (f_{O_1}, f_{O_2} and f_{O_3}). For inbound tunnels, A (receiving application layer messages) generates symmetric keys for protected communication with each of the inbound forwarding peers f_{I_1}, f_{I_2} and f_{I_3}. In both cases, A uses

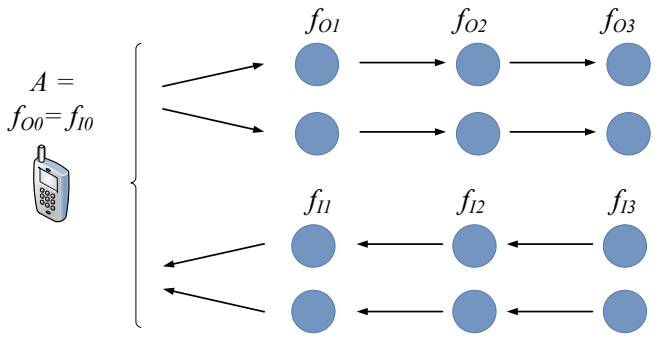


Fig. 8.2: Inbound and outbound tunnels of sender/receiver A

the public keys of the forwarding peers to distribute the required symmetric keys which will be used during the tunnel lifetime. Additionally, the forwarding peers establish TLS sessions for hop-by-hop security. Figure 8.3 sketches the resulting encryption and integrity-protection layers. The layered encryption ensures that the message looks different for each hop.

While the end-to-middle symmetric keys are valid only for the tunnel lifetime, a hop-by-hop TLS session may be multiplexed for several inbound and outbound tunnels serving several sender/receiver peers and can be long-lasting. This design decision is borrowed from Tor and should make traffic analysis more difficult. Unlike Tor where all peers are connected in a full mesh and establish TLS tunnels to each other, Pr^2-P2PSIP TLS tunnels are established on demand, since otherwise Pr^2-P2PSIP could not scale to more than few thousand peers.

Forwarding Pool

To discover forwarding peers, peers query the forwarding overlay. Additionally, each peer keeps a local pool of the forwarding peers it has learned about, and which it can ask to be a part of its tunnels. This pool should be kept up-to-date, so a peer can refresh its inbound or outbound tunnels.

The peer will occasionally learn about other forwarding peers as a side effect of overlay maintenance. However, it is crucial for the privacy goals of Pr^2-P2PSIP to not rely solely on overlay maintenance for re-filling its forwarding pool and not to simply choose peers from its overlay routing table. Instead, a UA A should perform node lookups (a FIND_NODE RPC in Kademlia) for *random* identifiers in the forwarding overlay when it needs to update its forwarding pool, in order to prohibit an attacker M from being able to force A to select her (M) as a forwarding peer in her tunnels (i.e., path selection attack; see Section 8.2.1).

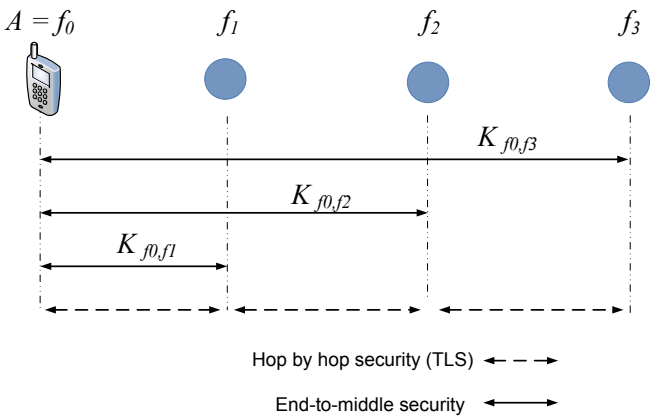


Fig. 8.3: End-to-middle and hop-by-hop encryption and data-integrity layers in Pr²-P2PSIP

8.1.2.2 Contact Data Storage

The contact data of all UAs are stored in a DHT. For each UA A there exists a value stored in the DHT with the contact data of A under the key $h(A)$. The contact data is a tuple $(+K_A, L(A,t))$. $L(A,t)$ does not reveal any information about A's real location $l(A,t)$. Instead, $L(A,t)$ includes information about the entry points of A's inbound tunnels (i.e., the forwarding peers furthest from A in her inbound tunnels), which will forward incoming messages towards A. Details on the structure of $L(A,t)$ are provided in Section 8.1.3.2.

8.1.3 Protocol Operations

In this section we provide more low level details on the protocol operations of Pr²-P2PSIP.

8.1.3.1 Tunnel Setup

Up to this point we have differentiated between inbound and outbound tunnels. However, the procedure for setting up both kinds of tunnels and the per-hop state required for them is the same. A forwarding peer can be unaware of the type of tunnel it is participating in. This reduces the complexity of Pr²-P2PSIP.

In fact, in both cases communication takes place in both directions, for instance to acknowledge tunnel setup and to tunnel RPC responses backwards to the initiator of a RPC (this is the case for publishing data in the DHT; see Section 8.1.3.2; and retrieving data from the DHT; see Section 8.1.3.3).

Forwarding peers need to store state information that is required to process incoming and outgoing messages for each tunnel. Let A be the UA which initiates the tunnel setup for sending or receiving application layer messages. Let f_1, f_2 and f_3 be the forwarding peers chosen by A to build the tunnel (as in Figure 8.3). A uses its pseudonym f_0 to communicate with the first hop in the tunnel, f_1. The state stored at each forwarding peer $f_i, i = 1, 2, 3$, called the *tunnel binding* in Pr^2-P2PSIP, is a tuple which consists of the following data:

- *tunnel ID*: a tunnel ID α used for multiplexing between different tunnels,
- *successor* and *predecessor*: the pseudonyms, public keys and locations of the successor and the predecessor peers in the tunnel: $(f_{i+1}, +K_{f_{i+1}}, l(f_{i+1}, t))$ and $(f_{i-1}, +K_{f_{i-1}}, l(f_{i-1}, t))$,
- *end-to-middle symmetric key*: K_{f_0,f_i}.

This data is distributed by A during the tunnel setup. Furthermore, f_{i+1} and f_{i-1} are used at each forwarding peer locally to determine whether it has already established TLS sessions with the successor and predecessor peers.

The data for the tunnel binding is sent by A onion-encrypted along the tunnel. For each node $f_i, i = 1, 2, 3$, A sends (indirectly) a message:

$$\begin{aligned} m_i = \quad & (\alpha, \\ & f_{i+1}, +K_{f_{i+1}}, l(f_{i+1}, t), \\ & f_{i-1}, +K_{f_{i-1}}, l(f_{i-1}, t), \\ & K_{f_0,f_i}) \end{aligned} \tag{8.1}$$

For f_3, the information about the successor is marked with *null* values:

$$\begin{aligned} m_3 = \quad & (\alpha, \\ & null, null, null, \\ & f_2, +K_{f_2}, l(f_2, t), \\ & K_{f_0,f_3}) \end{aligned} \tag{8.2}$$

Of course, this has the consequence that f_3 can deduce that it is the last hop in the tunnel. The impact of this information available to f_3 will be discussed in Section 8.2.1. The message flow for setting up the tunnel initiated by A looks as follows.

$$\begin{aligned} f_0 \leftrightarrow f_1 \quad &: \quad TLS \quad handshake \\ f_0 \rightarrow f_1 \quad &: \quad \{m_1, \{m_2, \{m_3\}_{+K_{f_3}}\}_{+K_{f_2}}\}_{+K_{f_1}} \\ f_1 \leftrightarrow f_2 \quad &: \quad TLS \quad handshake \\ f_1 \rightarrow f_2 \quad &: \quad \{m_2, \{m_3\}_{+K_{f_3}}\}_{+K_{f_2}} \\ f_2 \leftrightarrow f_3 \quad &: \quad TLS \quad handshake \\ f_2 \rightarrow f_3 \quad &: \quad \{m_3\}_{+K_{f_3}} \end{aligned} \tag{8.3}$$

The TLS handshakes take place only if two successive forwarding peers have not yet established a TLS session. After tunnel setup, A (i.e., f_0) can exchange messages with f_3 without revealing her location $l(A,t)$ or her identity (neither the public identity A, nor her pseudonym f_0). f_3 knows only the information about f_2.

A message m from A to f_3 is forwarded as follows:

$$\begin{aligned} f_0 \rightarrow f_1 &: \{\alpha, \{\alpha, \{\alpha, m\}_{K_{f_0,f_3}}\}_{K_{f_0,f_2}}\}_{K_{f_0,f_1}} \\ f_1 \rightarrow f_2 &: \{\alpha, \{\alpha, m\}_{K_{f_0,f_3}}\}_{K_{f_0,f_2}} \\ f_2 \rightarrow f_3 &: \{\alpha, m\}_{K_{f_0,f_3}} \end{aligned} \tag{8.4}$$

while a message from f_3 to A is forwarded as follows:

$$\begin{aligned} f_3 \rightarrow f_2 &: \alpha, \{m\}_{K_{f_0,f_3}} \\ f_2 \rightarrow f_1 &: \alpha, \{\{m\}_{K_{f_0,f_3}}\}_{K_{f_0,f_2}} \\ f_1 \rightarrow f_2 &: \alpha, \{\{\{m\}_{K_{f_0,f_3}}\}_{K_{f_0,f_2}}\}_{K_{f_0,f_1}} \end{aligned} \tag{8.5}$$

The tunnel setup (message flow (8.3)) is acknowledged by the last forwarding peer f_3. Thus, the acknowledgment message is the first message sent from f_3 to A via f_2 and f_1. Note that the acknowledgment of the tunnel setup by f_3 is crucial for the reliability of Pr²-P2PSIP. This will be discussed in detail in Section 8.2.2.

8.1.3.2 Publishing UA Contact Data

Publishing the contact data of a UA in the DHT makes use of outbound tunnels and the Kademlia STORE RPC. A UA A publishes its application layer public key ($+K_A$) as well as the pseudonyms, the public keys and the locations of the entry points of its inbound tunnels. For example, assume A has three parallel inbound tunnels. Then, the value stored in the DHT under the key $h(A)$ is a tuple $(+K_A, L(A,t))$ where

$$\begin{aligned} L(A,t) = \; & (f_{I_3}, +K_{f_{I_3}}, l(f_{I_3},t), \alpha), \\ & (f'_{I_3}, +K_{f'_{I_3}}, l(f'_{I_3},t), \beta), \\ & (f''_{I_3}, +K_{f''_{I_3}}, l(f''_{I_3},t), \gamma) \end{aligned} \tag{8.6}$$

where f_{I_3}, f'_{I_3} and f''_{I_3} are the entry points of the different inbound tunnels; and α, β and γ the respective tunnel IDs. The STORE RPC request is sent from A to f_{O_3} using message flow (8.4). This is depicted in Figure 8.4. It is crucial that the STORE RPC responses received by f_{O_3} are forwarded back to A (using message flow (8.5)). The reason for this is that A can not be sure that all peers in the outbound tunnel (f_{O_1}, f_{O_2} and f_{O_3}) are still online since the tunnel has been established or refreshed. If A does

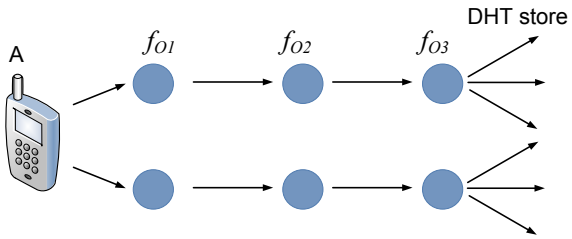


Fig. 8.4: Publishing UA contact data in the DHT

not receive a response to her STORE request from f_{O_3}, she needs to re-initiate the RPC using another outbound tunnel. The time interval between two successive RPC requests is a trade off between latency and signaling overhead. In the extreme case, A could send STORE RPCs simultaneously along several outbound tunnels. However, this parallelism may produce a large unnecessary signaling overhead depending on the stability of the network (and thus, the stability of the outbound tunnels). As a trade off, we use an aggressive timeout of $1s$ before the next outbound tunnel is invoked.

8.1.3.3 Retrieving Contact Data

Looking up data in the DHT is quite similar to publishing data in the DHT except the Kademlia RPC used is FIND_VALUE. A uses one of her outbound tunnels and asks the last peer in the tunnel to lookup the data on behalf of her. The same procedure with timeouts is performed if no response is received from an outbound tunnel.

Using the same procedure for publishing and retrieving data in/from the DHT reduces the complexity of the protocol.

8.1.3.4 Bidirectional Signaling

Once A has found the entry points of the inbound tunnels of B, she can use her outbound tunnels to send application layer messages to B. A may include her real location $l(A,t)$ (encrypted with $+K_B$) in the first signaling message to B or $L(A,t)$ if she does not want to reveal her location to B. The same holds for the response of B to A. Every SIP message is acknowledged end-to-end, i.e., if B receives a message from A through one of his inbound tunnels, he sends an acknowledgment through one of his outbound tunnels.

The same procedure with timeouts applies here as well: if A sends a SIP message to B and the acknowledgment does not reach A within $1s$ another end-to-end path, i.e., another combination of an outbound tunnel of A and an inbound tunnel of B is used. At this point, it is worth it mentioning that Pr^2-P2PSIP is designed with signaling in mind and is not optimized for real-time communication. The main problem with real-time communication is the accumulated one-way-delay in both directions between A and B, given

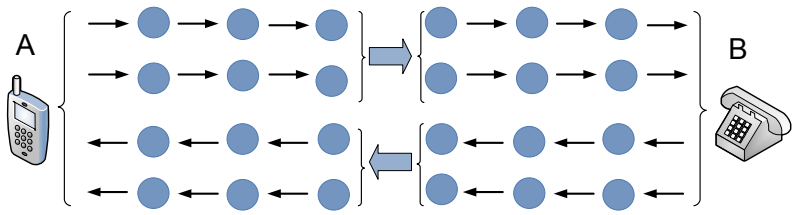


Fig. 8.5: Bidirectional signaling in Pr²-P2PSIP

that there are four to six hops between A and B (depending on the tunnel length).

8.1.4 Cryptographic Primitives

In this section, we provide implementation details on the cryptographic primitives used in Pr²-P2PSIP.

Symmetric Cryptography

As mentioned in Table 8.1, $\{m\}_{K_{a,b}}$ is a message m encrypted and integrity-protected with the shared key $K_{a,b}$. This is used to provide end-to-middle security in the inbound and outbound tunnels (see Figure 8.3). However, it is well known that different keys should be used for different purposes and for each direction [42]. Thus, four symmetric keys are derived from $K_{a,b}$ on both sides using a cryptographic key expansion function. These keys are derived at the tunnel setup and used during the tunnel lifetime.

Public Key Cryptography

Given that Pr²-P2PSIP makes extensive use of public key encryption, in particular for inbound and outbound tunnel setup, it is crucial to optimize the use of the public key cryptographic primitives. We use two solutions for this purpose:

- a message m from a to b encrypted with the public key $+K_b$ is actually encrypted with a temporary symmetric key $K_{a,b}$ generated by a. Then, $\{m\}_{K_{a,b}}$ is sent together with the temporary key $K_{a,b}$ encrypted with $+K_b$. Thus, $\{m\}_{+K_b}$ is actually implemented as $(\{K_{a,b}\}_{+K_b}, \{m\}_{K_{a,b}})$.

- an important design decision in Pr²-P2PSIP is to use Elliptic Curve Cryptography (ECC) [122] instead of RSA for public key encryption. The reason is the convenient key length without necessarily sacrificing performance. An ECC key length of 194 bits provides comparable

entropy to a 2054 bit RSA key[1].

The impact of the design decisions on the cryptographic primitives are further discussed in Section 8.2.3.

8.1.4.1 Pitfalls

In this section, we explain a few details that need to be taken into account when implementing Pr^2-P2PSIP. These details were skipped in the previous sections for the sake of simplicity.

Outbound tunnels used by A for publishing $L(A,t)$ should not be used for other purposes, e.g., retrieving contact data of another UA B. The last hop in the outbound tunnel of A, f_{O_3} sees only the hash value of A when the data is stored in the DHT. However, if f_{O_3} has a list of user names, it can determine whether A is one of them. If the same outbound tunnel is used for retrieving the contact data of B, f_{O_3} can deduce that A is about to send a SIP message to B. Thus, the social interaction privacy of A would be broken.

In the description of the tunnel setup in Section 8.1.3.1, the tunnel ID α remains constant along the tunnel. However, this raises a privacy threat especially for inbound tunnels. Intermediate hops (f_{I_1}, f_{I_2} and f_{I_3}) are all aware of the tunnel ID α published in the contact data of A in the DHT: $L(A,t)$. Thus, by crawling the DHT, f_{I_1} can discover which UA A has published its contact data $L(A,t)$ with α as tunnel ID, and can deduce the public identity of A. Since f_{I_1} has direct IP communication with A, the location privacy of A is broken. In order to defeat this attack, the tunnel ID has to be changed at each hop. Thus, each forwarding peer has two different tunnel IDs, one shared with the predecessor and another one shared with the successor. Since A needs to know the final tunnel ID at f_{I_3} in order to publish its contact data $L(A,t)$ in the DHT, f_{I_3} informs A about the tunnel ID to be published when it confirms the tunnel setup to A. Since f_{I_3} and A use end-to-middle encryption to secure their communication, f_{I_1} and f_{I_2} can not deduce which tunnel ID is published in the DHT.

8.2 Evaluating Pr^2-P2PSIP

8.2.1 Threat Analysis

In this section, we evaluate whether Pr^2-P2PSIP fulfills its goals, i.e., whether it can thwart attacks on location privacy and social interaction privacy. Additionally, based on an extensive threat analysis, we deduce appropriate recommendations for the tunnel length.

[1] The choice of the private key for RSA is limited by the choice of prime numbers, while any random number can be used as a private key for ECC.

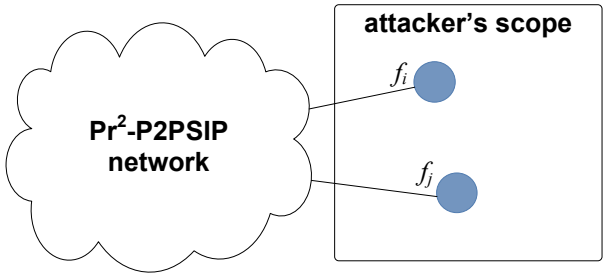


Fig. 8.6: Passive attacks on Pr^2-P2PSIP

The threat analysis of Pr^2-P2PSIP benefits from attacks on anonymization networks that have been described in the literature. Therefore, we provide an overview of those attacks that are relevant to Pr^2-P2PSIP first. We then evaluate whether these attacks can be applied to Pr^2-P2PSIP and if Pr^2-P2PSIP introduces new attack vectors.

8.2.1.1 Attacks on Anonymization Networks

Attacks on anonymization networks can be classified into *passive* and *active* attacks. Passive attacks are attacks where the attacker monitors communication between other peers. For this purpose, the attacker may try to become part of one of the victims tunnels. However, in passive attacks, attackers do not alter the data they observe or forward. In contrast to passive attacks, active attacks involve a participant actively altering or injecting data in the network. Nevertheless, an attacker may combine passive and active attacks in order to reach his malicious goals. As with all privacy preserving networks, a trade off exists between usability and security.

Traffic Analysis

Traffic analysis is a general term referring to monitoring data as it passes through a network to glean useful information. In an onion routing network over the Internet this typically means monitoring underlying network communications or data handled by a participant in the network overlay. A subset of traffic analysis called *timing analysis* measures *when* data enters or exits the network or nodes in the network. All of the attacks described herein utilize some form of traffic analysis.

As discussed in [9, 131] an attacker that is able to observe both ends of a tunnel may be able to correlate that two peers (identified by IP addresses) are communicating by analyzing inbound and outbound packet counts between every two peers. This attack is depicted in Figure 8.6. However, the attacker can not be sure that the two peers are communicating, since they could simply be forwarding data for other peers.

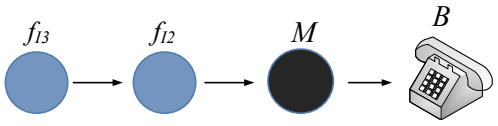


Fig. 8.7: Path selection attacks on Pr^2-P2PSIP

Path Selection Attacks

Another type of passive attack is the path selection attack [18]. The attacker forces particular peers to be chosen for a tunnel, preferably controlled by the attacker. Since we assume peers do not collude in Pr^2-P2PSIP, this attack is useful only if the attacker is on an end of the tunnel directly connected to the victim as in Figure 8.7.

Given that peers choose forwarding peers using *random* identifiers in the forwarding overlay, the probability of a successful path selection attack when a peer builds its inbound tunnels is inversely proportional to the size of the network. However, given that a peer occasionally has to change the peers in its inbound tunnel, the probability of a successful path selection attack grows over time.

Most other passive attacks [9,30] require a *global* passive adversary, outside of the threat model for our work.

Congestion Attacks

The congestion [92] or circuit clogging [87] attack combines typical traffic and timing analysis with an active denial or reduction of service attack. The basic layout of this attack is depicted in Figure 8.8. In this type of attack, a malicious peer initiates a "legitimate" communication with the victim. Using this communication, she alternates between periods of sending data and being silent on the tunnel. She concurrently builds tunnels between all (or some subset of) possible other peers in the network and sends probe traffic down each. If she can correlate the sending periods on the legitimate tunnel with traffic on the probe tunnels she has discovered that some peers on the probe tunnel are also part of the legitimate one. This method works if forwarding peers have to split resources equally between their tunnels; utilizing one tunnel therefore alters the latency properties of the other tunnels. By building repeated probe tunnels through different sets of possible peers she can eventually determine exactly which peers are being used. Provided that the peers on the legitimate tunnel are rotated over time (as is the case in Pr^2-P2PSIP) and the victim will be the only peer which will be always part of the tunnels, the attacker could discover the actual IP address of the victim.

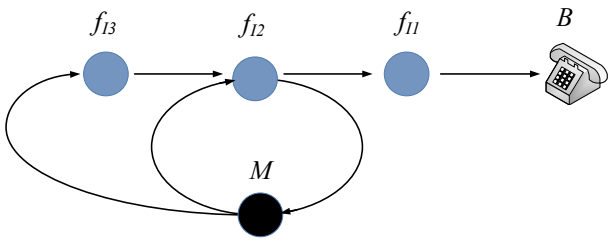


Fig. 8.8: Congestion attacks on Pr²-P2PSIP

8.2.1.2 Attacks on Pr²-P2PSIP

In this section, we provide a security threat analysis of Pr²-P2PSIP on inbound and outbound tunnels for different tunnel lengths.

Attacks on One-hop Inbound Tunnels

Using one-hop inbound tunnels, the only inbound forwarding peer and potentially malicious peer $M = f_{I_1}$ is directly connected to the victim B. The contact data of B published in the DHT points to f_{I_1}. Thus, by crawling the DHT (i.e. the storage overlay), f_{I_1} can find out which UAs have published their contact information with f_{I1} as a tunnel entry point. f_{I1} might be the tunnel entry point for several peers, let' say B, B' and B''. After collecting the data $\{L(B,t), L(B',t), L(B'',t)\}$ from the DHT, f_{I1} can correlate the tunnel IDs in $L(B,t)$, $L(B',t)$ and $L(B'',t)$ with the tunnel bindings it has previously setup and can unambiguously deduce the location of B, B' and B''.

Attacks on Two-hop Inbound Tunnels

Using two-hop inbound tunnels, as shown in Figure 8.9, a similar attack remains possible. A malicious peer M can trivially recognize from communication with the successor and the predecessor in the tunnel that she is not the entry point of the tunnel. Thus, M can deduce its position in the tunnel and that its predecessor is the initiator of the tunnel (B) and its successor is the entry point of the tunnel (f_{I_2} in Figure 8.9). By crawling the content of the DHT, M can find out which UAs have published their contact information with f_{I_2} as a tunnel entry point, again let' say B, B' and B''. The difference to the one-hop case is that M can not necessarily identify which one of these peers is the initiator of the tunnel she is part of. This is because the tunnel ID is not constant along the tunnel. Nevertheless, M could significantly reduce the number of possible public identity of the tunnel initiator, potentially to one. This would lead to an unambiguous link between the public identity of B and his current location $l(B,t)$.

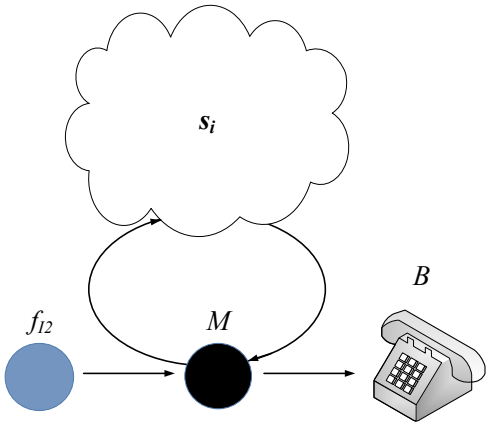


Fig. 8.9: Attacks on two-hop inbound tunnels

Depending on the size of the network, B may have changed its inbound tunnels while M is still crawling the DHT, and the data M is looking for in the DHT may become unavailable. However, we can not rely on this assumption, if M has sufficient resources.

Entry guards: One possible approach to reduce the probability of this attack could be the concept of *entry guards* [97], which were suggested for thwarting attacks on discovering the origin of *hidden services* in Tor. These attacks are based on path selection attacks. The concept of entry guards is as follows; instead of choosing uniformly at random from the set of all peers for the crucial hop (the nearest to the hidden server in Tor, the nearest to the UA in the inbound tunnel in Pr^2-P2PSIP, i.e., f_{I_1})), a small set of peers are chosen initially and one of these is always utilized in that position. Choosing forwarding peers uniformly at random gives a patient attacker the chance to be chosen as the crucial hop with a high probability if B rotates his tunnels regularly, whereas the probability of choosing the attacker with "final guardians" is only g/n where g is the total number of guardian nodes used (and n the overall number of peers as mentioned in Section 8.1.1).

Nonetheless, since malicious peers have the chance here to discover the public identity of B and its location with an effort estimated by $O(n)$ (crawling the DHT), we consider the attack on one-hop and two-hop inbound tunnels as a real threat to Pr^2-P2PSIP.

Attacks on Three-hop Inbound Tunnels

Using three-hop inbound tunnels, a possible attack scenario is a variant of the circuit clogging attack, where the participants of a tunnel can be deduced. In this scenario the attacker M initiates a communication with the victim B (Figure 8.8). M wants to discover the IP address of B. To

do so, she actively builds tunnels through many peers which she uses to send a steady stream of data to herself. She then sends a certain pattern to B (for example, via chat), which can be detected on the tunnels that she is monitoring because of interference [87, 92, 113]. Since M may not necessarily obey to the agreed inbound tunnel length in the network, she could conceivably connect to every peer with a one hop tunnel back to herself and send the pattern to B (via his legitimate inbound tunnel). If the pattern is detected, this reveals either B or a part of his tunnel. By repeating the same procedure for each of B's multiple inbound tunnels, M can eliminate B's tunneling peers, because B will be the only peer present on *each* of the inbound tunnels used.

This attack becomes more difficult as the number of peers in the network increases, because the attacker needs to monitor them all for the pattern she is sending. False positives or false negatives may occur due to other traffic in the network at the same time as the attacker's probe or pattern traffic. The attack may also take a prohibitively long amount of time to mount; if the attacker cannot monitor all nodes in the network at once, she will need to perform this attack by monitoring only some subset of the network at a time.

General Attacks on Outbound Tunnels

No matter how long the outbound tunnel is, the last hop in the tunnel (furthest from A) which is used for publishing the contact data of A in the DHT should not be used for other purposes as mentioned in Section 8.1.4.1. Otherwise, the social interaction privacy of A would be broken.

Attacks on One-hop Outbound Tunnels

If the outbound tunnel of a UA A consists of one hop only, when A publishes her contact data in the DHT, the outbound forwarding peer f_{O_1} receives the STORE RPC from A directly, and thus, can trivially discover the public identity of A and correlate it with her IP address. This would break the location privacy of A.

Attacks on Two-hop Outbound Tunnels

Attacks on two-hop outbound tunnels become more difficult. The last peer in the outbound tunnel f_{O_2} may misuse the property of Pr^2-P2PSIP that communication in both inbound and outbound tunnels takes place in both directions, and send certain traffic patterns to f_{O_1} which are forwarded to A and thus may be the basis for a congestion attack.

Conclusions

Given the threat analysis above, we conclude that:

- Passive attacks are of limited use because while they may reveal that two peers are participating in the network and connected, this does not indicate whether the peers are forwarding data for other peers or actually communicating.
- Path selection attacks require that the attacker be chosen as the victim nodes final inbound hop. The probability of the success of such an attack is inversely proportional to the size of the network. But it increases over the time by changing the tunnel. Unless entry guards are chosen as crucial hop.
- Congestion attacks may be feasible, but at high cost, take a long time and are susceptible to false positives and false negatives.
- A tunnel length of *three hops* for *inbound tunnels* and *two hops* for *outbound tunnels* provide location and social interaction privacy at a high and satisfactory degree.

8.2.2 Reliability Cost Analysis

In this section, we provide a model of Pr^2-P2PSIP based on reliability theory [111]. This model will then be used for estimating the overhead generated by adding privacy to CoSIP or P2PSIP. Note that the reliability analysis below requires some background knowledge on reliability theory introduced in Chapter 2, Section 2.1. Furthermore, the reliability analysis of non-privacy CoSIP provided in Chapter 4, Section 4.4.1 is also useful to understand the reliability analysis for Pr^2-P2PSIP below.

8.2.2.1 Modeling Pr^2-P2PSIP Networks with Reliability Theory

A Pr^2-P2PSIP network is a system which consists of multiple units, which are the peers. The time to failure of a peer is the time interval between the time when the peer goes online until it leaves the network, i.e., T is the peer lifetime.

Reliability Model of Pr^2-P2PSIP

A UA B refreshes his contact data in the DHT as well as his inbound tunnels periodically with a refreshing period e.g., $\tau = 20mn$, in order to make sure it remains reachable in the Pr^2-P2PSIP network with high probability. This high probability is a target reliability, e.g., $\bar{R} = 1 - 10^{-5}$.

Furthermore, in the reliability analysis of non-privacy-preserving CoSIP (Chapter 4) we introduced the value μ as being the minimum reliability of a

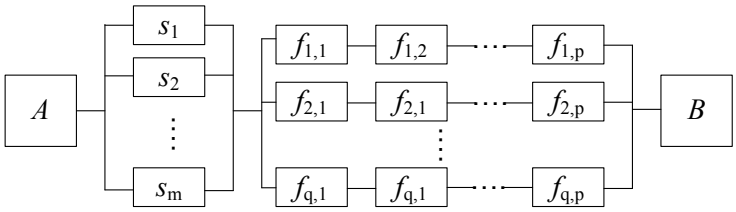


Fig. 8.10: Reliability model of Pr²-P2PSIP

single peer at the end of the refreshing period τ (See Figure 4.10). The value μ is significant for the reliability analysis below, since the overall reliability of Pr²-P2PSIP needs to remain above or equal to $\bar{R}$ even if the reliability of single peers drops down to $\mu < \bar{R}$.

Figure 8.10 shows the resulting reliability model of Pr²-P2PSIP. A UA A calling B needs to reach at least one of the storage peers s_i which have stored the contact data of B. Then, A needs to find at least one inbound tunnel to B where all peers which build the tunnel are still online. As shown in Figure 8.10, let m be the number of storage peers, p the length of B's inbound tunnels and q the number of parallel inbound tunnel.

Estimating the Overhead of Privacy

If $p = 0$, then we have a regular P2PSIP network. Let m_0 the number of required parallel storage peers, then it follows from equation (2.5):

$$1 - (1 - \mu)^{m_0} \geq \bar{R} \tag{8.7}$$

Thus, the number of required storage peers for an inbound tunnel length $p = 0$ can be estimated by:

$$m_0 \geq \frac{log(1 - \bar{R})}{log(1 - \mu)} \tag{8.8}$$

If $p \geq 1$, then the reliability of the storage part at the end of each refreshing period can be estimated as:

$$(1 - (1 - \mu)^m) \tag{8.9}$$

and the reliability of the inbound forwarding part:

$$(1 - (1 - \mu^p)^q) \tag{8.10}$$

Let $\bar{R}_s$ the target reliability of the storage part and $\bar{R}_f$ the target reliability of the inbound forwarding part. Thus, m and q can be estimated as follows:

$$m \geq \frac{log(1 - \bar{R}_s)}{log(1 - \mu)} \tag{8.11}$$

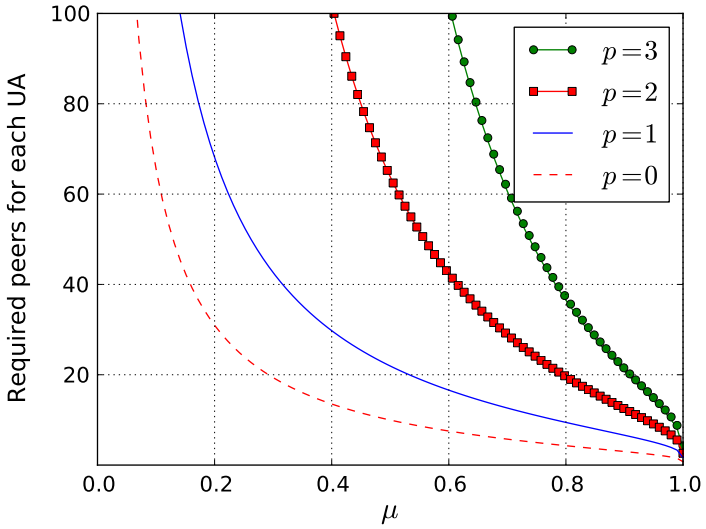


Fig. 8.11: Number of peers required to keep a UA reachable in a Pr²-P2PSIP network with target reliability $\bar{R} = 1 - 10^{-5}$

$$q \geq \frac{log(1 - \bar{R}_f)}{log(1 - \mu^p)} \tag{8.12}$$

and the reliability of the whole system:

$$(1 - (1 - \mu)^m).(1 - (1 - \mu^p)^q) \geq \bar{R}_s \bar{R}_f = \bar{R} \tag{8.13}$$

As it can be seen in Figure 8.10, the overall number of peers required for each UA in order to be reachable is $(m + pq)$.

By varying the ratio $\bar{R}_s/\bar{R}_f$ for a constant system target reliability $\bar{R} = 1 - 10^{-5}$ we obtain different values for $(m + pq)$ which are slightly better than equal target reliabilities for both parts, i.e., $\bar{R}_s/\bar{R}_f = 1$. Thus, we determine numerically the optimum value of $(m + pq)$ by varying $\bar{R}_s/\bar{R}_f$ for different values $p \in \{0, 1, 2, 3\}$ and $\mu \in (0, 1]$ and $\bar{R} = 1 - 10^{-5}$ (values of μ are chosen stepwise with steps of 0.01). Figure 8.11 shows the result. The number of peers required for a UA to be reachable for incoming SIP message increases to infinity if $\mu \to 0$ (i.e., average peer lifetime is $\varepsilon \to 0$) and converges to $(p + 1)$ for $\mu \to 1$ (i.e. a static network with peers never leaving). Figure 8.12 shows the relative overhead compared to a non-privacy-preserving P2PSIP network ($^{m+pq}/_{m_0}$). The relative overhead for $p = 1$ is constant ($(^{2log(1-\sqrt{1-10^{-5}})}/_{log(10^{-5})}) \approx 2.12$) which is due the equal parts for storage and forwarding in that case.

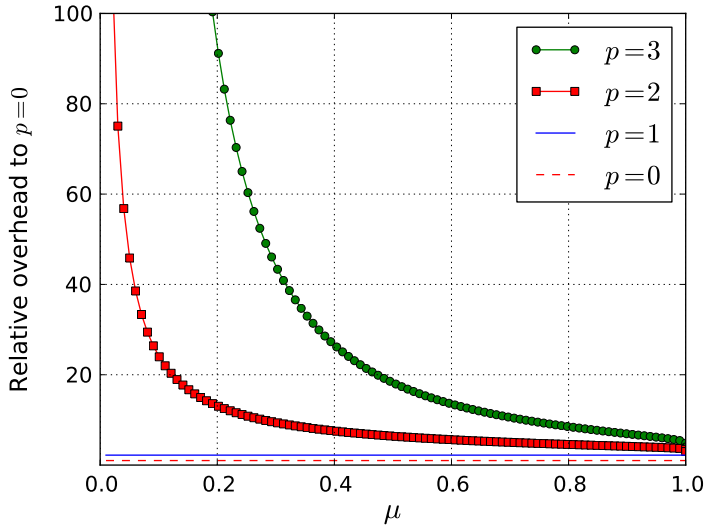


Fig. 8.12: Overhead of privacy compared to a regular P2PSIP network.

Interpretation based on Skype Traces

Using the Skype network as an example, according to [48], around 87% of the Skype super-peers have a peer lifetime more than $30mn$ and 78% more than $1h$. We interpolated these values to estimate the privacy overhead for $p = 3$ with different refreshing periods. The result is shown in Table 8.2. E.g., assuming a refreshing period of $20mn$ in Pr²-P2PSIP, then around 33 peers would be required to keep a UA reachable for incoming calls. However, taking only Skype *super-peers* into consideration means that in Pr²-P2PSIP only stable peers should be used for storage and inbound tunnels.

Note that if a UA needs around 33 peers for storage and inbound tunnels, this means also that each UA will receive on average 33 requests within $20mn$

Tab. 8.2: Estimation of the privacy overhead based on Skype traces

Refreshing period (τ)	μ	Number of storage peers (m)	Number of inbound tunnels (q)	Total number of peers ($m + pq$)
10.0 mn	0.95	5	7	26
20.0 mn	0.91	6	9	33
30.0 mn	0.87	6	12	42
40.0 mn	0.84	7	14	49
50.0 mn	0.81	8	17	59
60.0 mn	0.78	9	19	66

from other peers to store data or be a part of an inbound tunnel. Additional signaling is required for the outbound tunnels, overlay maintenance and DHT lookups.

Conclusions

The reliability analysis above provides an estimation of the impact of adding privacy to P2PSIP. The signaling overhead generated by Pr^2-P2PSIP to keep a target reliability of "5 nines" should not be underestimated. Further, the overhead is sensitive to the stability of the storage and forwarding peers. This may have different consequences depending on the types of devices used for the UAs. Processing a few requests per minute for storage, tunnels, DHT lookups and overlay maintenance may not be a problem for fixed hardphones, but would mean a large resource consumption for mobile devices, in particular if they are constantly awoken from standby mode (at least, this is a problem today). Given that the signaling overhead is sensitive to the stability of the storage and forwarding overlay networks, it is crucial for Pr^2-P2PSIP to exclude peers with a short lifetime from these overlays.

It is well known that P2P networks perform better under a low rate of churn and that a hierarchy with super peers should be introduced if the churn rate becomes higher. However, as it can be seen in Figure 8.12, these facts can reach other dimensions under privacy constraints.

8.2.3 Cryptographic Overhead

Given the design decisions described in Section 8.1.4, the overhead of the public key encryption of a message m sent from a to b using a 194 bit ECC key $+K_b$ and a 128 bit temporary symmetric key $K_{a,b}$ for AES encryption in CBC mode consists of:

- the length of $\{K_{a,b}\}_{+K_b}$, which results in an ECC block size of 194 bits,
- the length of the initialization vector used for the symmetric encryption in CBC mode: 128 bits,
- and a maximum padding of 128 bits for the symmetric encryption,

which results in an overall overhead between 322 and 450 bits, i.e,. approximately between 40 and 56 bytes. Thus, even if a message is onion-encrypted with three layers the overhead in terms of message length remains acceptable.

However, the cryptographic overhead of Pr^2-P2PSIP in terms of the number of public key operations increases linearly with the number of tunnels per UA and the number of peers per tunnel. Thus, the same conclusions hold here as in Section 8.2.2.

8.2.4 End-to-end Signaling Latency

The signaling latency from UA A to UA B is affected by:

1. the processing overhead at each forwarding peer,
2. the tunnel length, or the number of forwarding peers used for inbound and outbound tunnels,
3. the accumulated one-way-delay along the full path between A and B,
4. the probability that all forwarding peers in a path are online since they were last.

As mentioned in Section 8.1.4, once a tunnel is setup, only symmetric cryptography is used. Thus, the cryptographic processing is certainly not a bottleneck. As for the tunnel length and the accumulated delay, we believe that Pr^2-P2PSIP deployed with the recommended tunnels lengths in Section 8.2.1 does not necessarily involve more signaling hops than server-based SIP networks used in practice today, in particular, where quite a few components are involved in the signaling for different purposes, e.g., lawful interception, billing, etc.

As for the probability that all forwarding peers in a path are online, as mentioned in Section 8.1.3.4, A tries another end-to-end path, i.e., another combination of outbound tunnel of A and inbound tunnel of B if it does not receive an acknowledgment to a SIP message within $1s$.

Thus, the maximum overall signaling latency is expected to be within a few seconds. If peers in the forwarding overlay are stable, it becomes more likely that the tunnels are available and the signaling succeeds at the first attempt, thus reducing the latency by an order of magnitude. If Pr^2-P2PSIP is used for chat, the same tunnels should be used for subsequent chat messages, since once tunnels have been successfully used, they are likely to remain available for the next chat messages, assuming a heavy-tailed distribution of the peer lifetime.

8.3 Related Work

Location privacy was not a main concern when the Internet was conceived, because hosts were fixed. However, it was considered early on in GSM standardization. In GSM and UMTS networks, each mobile devices has a unique identifier called the International Mobile Subscriber Identity (IMSI). However, temporary pseudonyms called Temporary Mobile Subscriber Identities (TMSI) are usually used for communication with base stations. Nevertheless, both GMS and UMTS authentication protocols allow an attacker to impersonate a base station and request the User Equipment (UE) to send its IMSI for authentication.

Seedorf [128] discusses the security issues inherent in P2PSIP and mentions privacy briefly. In [12], the authors investigate a game theoretical approach for the security threats of P2PSIP such as SPIT and attacks on overlay routing. However, privacy is not addressed.

RELOAD [61], the base protocol for P2PSIP allows for different overlay algorithms to be plugged in. The IETF P2PSIP WG charter [98] does not preclude the deployment of anonymization networks. However, it can not be assumed that any general purpose anonymization network could be used. The Internet draft [60] describes SIP usage for RELOAD and mentions explicitly that "all RELOAD SIP registration data is public. Methods of providing location and identity privacy are still being studied". Thus, Pr^2-P2PSIP is right on target to address this issue. In [155], the authors investigate the costs of maintenance and lookup in DHTs with different ratios of super peers. Their work considers regular DHT functionality without privacy. Nonetheless, our work can be enhanced in the future with a similar analysis in order to provide better insight on the signaling overhead of Pr^2-P2PSIP with different ratios of fixed and mobile devices with different resources. In [27, 151] the authors demonstrate how the end points of P2P VoIP streams, e.g. Skype streams, can be identified. Thus, they demonstrate how one could break location and social interaction privacy. However, Skype peers do not consider each other as potentially malicious.

There are many anonymization networks which utilize onion routing [45] or a derivative, notably Tor [35], JAP [40], MorphMix [113] and I2P [39]. They all share characteristics and sometimes differ only in subtle ways. Our intention is not to invent a new anonymization network or new anonymization techniques, but to leverage existing techniques, particularly onion routing and inbound and outbound tunnels to address the privacy issues of P2PSIP. Nevertheless, Pr^2-P2PSIP can still be clearly differentiated from existing anonymization networks in several aspects. Approaches for anonymization networks can be classified into centralized and P2P approaches. Pr^2-P2PSIP is a P2P approach. Centralized approaches, e.g., Tor [35], Crowds [112] and MorphMix [113] rely on centralized databases (although eventually redundant as in the Tor case) to get a list of relay nodes. Pr^2-P2PSIP relies on a forwarding overlay. Likewise, Tor hidden services, which can be compared to Pr^2-P2PSIP inbound tunnels, are accessed via service descriptors stored in a central database. In Pr^2-P2PSIP, peers get the contact data from the DHT before they contact the inbound tunnel entry points.

In P2P anonymization networks, such as I2P [39], Salsa [93], Cashmere [154], Tarzan [43] and AP3 [89], there is no central authority as in Pr^2-P2PSIP, which renders them vulnerable to Sybil attacks. Further, peers select forwarding peers from their P2P routing tables. This makes them vulnerable to attacks where malicious peers attempt to dominate the routing tables of other peers. Pr^2-P2PSIP uses a separate overlay for forwarding and chooses forwarding peers randomly.

Pr^2-P2PSIP allows anonymous routing only within the network. Other anonymity networks such as JAP [40], Cashmere [154], Tarzan [43], MorphMix [113] and Crowds [112] are designed to allow communication with normal servers in the Internet. Thus, they need to support outbound connections. On the other hand, the clients do not have to be reachable for incoming communication as in Pr^2-P2PSIP.

In summary, Pr^2-P2PSIP benefits from the design of Tor and other anonymization networks and experience learned from them, while it has been designed exclusively to provide the P2P-based SIP user registration and session establishment, while preserving the privacy of the network participants. To the best of our knowledge, there has been no work which provides a dedicated solution to the privacy needs of P2PSIP with such an extensive analysis of the implications.

8.4 Conclusions

This chapter presents a feasibility study of adding privacy to CoSIP. While the solution Pr^2-P2PSIP has been motivated by the privacy concerns in CoSIP, its applicability is valid for both CoSIP and the IETF P2PSIP. Pr^2-P2PSIP provide adequate location privacy and social interaction privacy. Is benefits from the design of well known anonymization networks, such as Tor and I2P. In Pr^2-P2PSIP peers use outbound and inbound tunnels as well as pseudonyms to hide their locations and public identities. We provided an extensive threat analysis and estimated the costs of adding privacy to P2PSIP or CoSIP in terms of cryptographic overhead, signaling latency and reliability costs.

Our conclusions are as follows: Pr^2-P2PSIP provides location and social interaction privacy with a tunnel length of three for inbound tunnels and two for outbound tunnels. Cryptographic overhead is not a hindrance for Pr^2-P2PSIP, in particular if ECC is deployed. Signaling latency improves as the forwarding overlay becomes more stable. The signaling overhead to keep a target reliability of “5 nines” should not be underestimated. Further, the signaling overhead is sensitive to the stability of the forwarding overlay. Thus, it is crucial for a successful deployment of Pr^2-P2PSIP that stable peers, i.e., those with a long lifetime, are preferentially chosen for building tunnels and storing contact data.

Part IV

CONCLUSIONS AND OUTLOOK

9. CONCLUSIONS AND OUTLOOK

This thesis mainly connects three research themes; network security, network reliability and P2P networking. Network resilience is an emerging discipline which encompasses both network security and reliability. VoIP is the main application used to validate the proposed mechanisms. In this concluding chapter, we summarize the achievements of this thesis and discuss trends and future directions.

9.1 Summary

We identified P2P networks as a potential resilience mechanism for application layer signaling. The reasons are the inherent decentralization, data replication, autonomous recovery from stale routing table entries, and geographic diversity. However, given the security issues in pure P2P networks, we followed a supervised P2P network approach to enhance the resilience of the signaling. In a supervised P2P network, an overlay supervisor is involved in forming the overlay. However, the failure of the network supervisor does not lead to the failure of the service provided by the overlay.

We presented CoSIP (Chapter 4), which is an implementation of the supervised P2P network approach in the SIP context. In CoSIP, SIP endpoints organize themselves in a DHT. User registration occurs in parallel at the SIP server and in the P2P network. In case of server failure, the Caller can still establish phone calls by retrieving the Callee contact data from the DHT. We have shown based on reliability theory and traces from the Skype network that the P2P network can enhance the signaling reliability by "3 nines" or "5 nines" with a small number of replica nodes (6 for "3 nines" and 10 for "5 nines"). Therefore, CoSIP server downtimes can be bridged with high probability. Moreover, CoSIP provides the benefit of geographic diversity of the peers (plus server) and potentially diversity in their software and hardware as well. Thus, the reliability of CoSIP is expected to be significantly higher than the reliability of a pure server solution.

On the other hand, SIP endpoints acquire verifiable identities from the CoSIP server (the supervisor) upon successful user authentication. This improves the security compared to pure P2P networks. A SIP UA acquires one verifiable identity at the application layer in form of a SIP URI embedded in an X.509 certificate. In contrast to current SIP networks where the SIP endpoints authenticate themselves only to SIP proxies and registrars, they

can use the certificates for P2P authentication among each other.

The second verifiable identity issued by the supervisor is used at the overlay layer. It is a node ID chosen by the supervisor and embedded in an X.509 certificate. It allows for proving the correctness of the node ID to other peers in the overlay and provides protection against Sybil, eclipse and chosen-location attacks. We implemented CoSIP as a proxy compatible with standard SIP UAs, using the Kademlia DHT and SER as a SIP server. We validated our implementation on local testbeds as well as PlanetLab and have shown that SIP UAs can establish phone calls both under normal operation and under SIP server failure.

In Chapter 5 and 6, we explored the design flexibilities in DHTs, notably with the supervised P2P network approach in mind. In particular, in Chapter 5 we questioned the efficiency of consistent hashing, the method used by most prominent DHT algorithms (Chord, Pastry, Kademlia) for assigning node IDs and data items to nodes. Consistent hashing suggests uniformly distributed random node IDs. This results into some peers being responsible for $O(log(n) \cdot 1/n)$ of the address space, n being the number of nodes in the overlay. We showed that the presence of an overlay supervisor which is responsible for assigning IDs to nodes guarantees a near-optimal node ID distribution where each node is responsible for $2/n$ of the address space at most. This reduces the expected maximum load from a logarithmic to a constant factor leading to an improved load distribution. We provided exact formula for the load distribution for *i)* the random node ID case and *ii)* the near-optimal case proposed in this thesis.

In Chapter 6, we questioned another myth about DHTs, namely that routing takes $O(log\, n)$ steps in average. We explored the option of reducing the routing complexity by increasing the routing table size. We provided an algorithm for building structured overlays with routing within $O(1)$ and routing table size in $O(\sqrt{n})$. More precisely, given random node IDs, routing can be achieved within two hops with a probability $(1 - \epsilon)$ for an arbitrarily small ϵ. Then, the approach with supervised node IDs from Chapter 5 is applied here as well to reduce the probability that routing within two hops fails to zero.

The security threat analysis in Chapter 7 has shown that verifiable IDs are not sufficient to address the security issues in P2P networks. For example, a malicious peer can still discard lookup messages, or forward them to random peers. Likewise, a malicious peer can still discard data items it is supposed to store or refuse to deliver them upon lookup requests. These attacks result in reducing the availability of the SIP signaling running on top of the overlay. Thus, security in P2P networks needs to be enhanced with redundancy in routing and storage. In particular, parallel lookups increase the probability to successfully route beyond faulty nodes. Iterative lookups allow for efficient detection of faulty nodes leading to a reduction of the lookup latency and an increase of the lookup availability.

The security threat analysis left two security threats unsolved, which result from using the P2P network for VoIP signaling. These threats are SPIT and attacks on privacy, notably location privacy and social interaction privacy. Concepts for SPIT prevention can be deployed only with limitations in a P2P environment. Therefore, it is recommended to allow for P2P signaling in CoSIP only with previously known UAs. Thus, in case of server failure, users can, e.g., still reach their families and friends using the P2P network. On the other hand, this confirms the CoSIP concept where the SIP infrastructure should be used for session establishment under normal operation and the P2P network is used for enhancing reliability but not as the only signaling solution.

Finally, attacks targeting user privacy are valid for CoSIP as well as the IETF P2PSIP. We used concepts and experiences learned from anonymization networks such as Tor and I2P and proposed Pr^2-P2PSIP, a privacy-preserving P2PSIP protocol (Chapter 8). An extensive analysis of Pr^2-P2PSIP in terms of cryptographic overhead, signaling latency and reliability has shown that CoSIP and P2PSIP can be enhanced with privacy. However, the signaling effort to keep a minimum target reliability of "3 nines" or "5 nines" becomes higher. This is especially the case if the P2P network has a high churn rate. Thus, it is critical for the efficiency of Pr^2-P2PSIP to use only stable peers for the anonymization or storage service in the P2P network.

9.2 Future Directions

The promising results of this thesis motivate future research directions.

CoSIP

Applications: Our CoSIP architecture can be extended to support further applications, e.g., virtual presence, IM and conferencing. The signaling for these services differs from the user registration and VoIP session establishment implemented and analyzed so far for CoSIP. Further applications are the discovery of nodes with special functions in the network, e.g., relays to forward traffic in case of lack of IP connectivity, PSTN gateways or gateways to other SIP domains.

Performance Evaluation: The feasibility study of adding privacy to CoSIP and P2PSIP has been promising. An evaluation with an implementation can certainly provide new ideas for performance optimization and reducing the signaling overhead.

Supervised P2P Networks

P2P networks are expected to benefit from a supervised approach notably in terms of performance optimization. We proposed and evaluated optimization in load balancing. Routing optimization in the context of low-diameter overlays proved to be feasible as well. It is expected that "regular" DHTs with $O(log\, n)$ routing can benefit from a supervised approach as well.

Recent research activities focused also on P2P networks with a server which provides joining peers with locality information, e.g., a list of other peers within the same autonomous system [5, 153]. The goals are improved latency from end-user perspective and the reduction of cross-domain traffic from network provider perspective.

Generally speaking, the benefit of a supervised approach compared to a pure P2P approach is that the supervisor disposes of a global overview of the network. The supervisor can be also a trusted third party for security purposes as in CoSIP.

There is another area where a supervised P2P network approach can be beneficial to enhance resilience: network connectivity. Peers should be able to provide connectivity to each other in case of infrastructure failures. For example, in mobile networks, end devices can establish connectivity among each other in case of a disaster where the network provider infrastructure becomes unavailable.

Directions in P2PSIP Server Architectures

Using a P2P architecture at the SIP server infrastructure side, VoIP providers can benefit from advantages inherent in P2P networks, in particular autonomous configuration, data (state) replication as well as recovery from partial failures. Moreover, P2P networks on the infrastructure benefit from a less stringent threat model since peers are considered trustworthy.

Nevertheless, a holistic resilience concept is still required. An efficient and well organized P2PSIP network at the infrastructure side may be deployed. However, if it is reachable via a single proxy, load balancer or NAT, this reduces the resilience of the whole system significantly. The system is only as resilient as the weakest link in the service chain. In VoIP, the chain consists of the components which are necessary to establish a session between the Caller and Callee. Thus, CoSIP provides an optimal solution from this perspective, since a Caller has multiple parallel paths to reach a Callee. An example where the service chain fails in a spectacular way is the large scale failure of the T-Mobile network mentioned in Chapter 4. The user database is implemented in a DHT with sophisticated resilience mechanisms. But a software update in the authentication server (HLR) rendered the user database unavailable, leading to nearly all T-Mobile subscribers in Germany (around 40 million) not able to make phone calls or send text

messages for four hours.

In Chapter 2, we discussed different emerging technologies for avoiding such single points of failure. DNS is one of the simplest ones, since DNS can provide multiple IP addresses in a response to a query with a single name. However, using DNS to provide multiple SIP proxies (as is already the case for web and video streaming) is currently not a common practice. This is the first single point of failure in current SIP architectures.

IP anycast is another technique which has been successfully deployed to enhance the resilience of DNS itself (See Section 2.3.1). Thus, IP anycast is a potential technique to enhance the resilience of SIP infrastructures as well. IP anycast can be used for failover from one SIP proxy to another. It allows for geographic diversity, and therefore provides better survivability in case of disasters.

Another emerging technology which has received a lot of attention in the last few years is *Cloud Computing*. In SIP infrastructures deployed in the cloud, new server instances can be started "on demand" if the load increases. They can join the server P2P network and do not require manual configuration.

Finally, the interaction between these components; the cloud, DNS, IP anycast, IP routing protocols, e.g., BGP, etc. needs to be carefully analyzed to provide a resilient and survivable VoIP signaling service. Otherwise, server-based systems remain prone to configuration faults and catastrophic failures.

Conclusions

The network and service landscape has been evolving continuously. Trends and techniques such as Cloud Computing and IP anycast offer a wide range of interesting research questions and ideas how to use them for enhancing network and service resilience, notably VoIP resilience. Supervised P2P networks allow for better performance optimization and can be used for further network resilience purposes, e.g., to enhance network connectivity.

Finally, given that the resilience requirements on the Internet keep on increasing, it can be expected that research topics around network resilience will remain interesting in the next years.

APPENDIX

A. COSIP PROTOTYPE IMPLEMENTATION DETAILS

In this chapter, we describe our implementation of CoSIP.

A.1 Design Decisions

On the main decisions is to leverage existing software components where possible and be compliant with standard SIP clients.

A.1.1 Support of Different SIP Clients Software

We implemented CoSIP as a local SIP proxy that processes the SIP signaling of one or more SIP UAs. Implementations of the SIP UA do not need to be aware of CoSIP. The SIP UA just needs to be configured with the CoSIP proxy as an outbound proxy.

A.1.2 Choice of a DHT

Our CoSIP implementation supports pluggable DHT implementations.

Bamboo:

Our CoSIP implementation supports the *Bamboo* DHT [114]. Bamboo uses the concept of the Pastry DHT [123] with improved routing and maintenance algorithms in order to cope with high churn rates. Bamboo needs less traffic for the overlay maintenance. For the communication between the CoSIP proxy and the DHT node, we use XML-RPC for performing `STORE` and `GET` requests. The XML-RPC interface is provided by Bamboo to simplify the integration of Bamboo into other projects.

Kademlia:

Additionally, CoSIP is interoperable with the *Kademlia* DHT [86]. We integrated our CoSIP implementation with the Kademlia implementation *entangled*[1].

One of the main motivations for using Kademlia is that it has been the only DHT algorithm which is actually used in practice. More precisely, the

[1] `http://entangled.sourceforge.net/`

KAD P2P network [140] is based on Kademlia. The Kademlia protocol is based on four RPCs only `FIND_NODE`, `FIND_VALUE`, `STORE` and `PING`.

A.1.3 A Supernode Approach

It is well known that due to the different capacities of peers in a P2P network, such as CPU, memory and connectivity, there should be a differentiation between "powerful" peers, which are called the Supernodes, which are connected to the DHT, and "weak" peers that are connected indirectly to the DHT via the Supernodes. This contributes to the stabilization of the DHT. In CoSIP, we support the Supernode approach by having several SIP UAs connecting to a CoSIP proxy. The CoSIP proxy deals with a few SIP UAs and represents them in the P2P network. However, SIP UAs may also connect to several CoSIP proxies at a time in order to avoid that the CoSIP proxy becomes a single point of failure by itself.

A.1.4 Putting Everything Together

Based on the design decisions presented above, the architecture results into different software components communicating together as presented in Figure 4.6. The SIP UAs use standard SIP to communicate with the SIP proxies. They are unaware of the use of CoSIP or the DHT. The CoSIP proxies use SIP to communicate with the SIP server and among each other. As a SIP server, we use the SIP Express Router (SER) [134]. The CoSIP proxy was implemented in Python.

We successfully tested our CoSIP proxy implementation with SER as a server, and Kphone [139], Ekigacite [57] on Linux as well as XLite [56] and QuteCom (former WengoPhone) [106] on Windows.

A.1.5 Support of P2P-based SIP

As a side effect of our implementation of CoSIP, it is easy to configure the functionality of CoSIP to support a P2P mode without a SIP server. This purely P2P mode can be used, e.g. in small networks where the social contact between the users may make a central authority unnecessary and, e.g., self signed certificates may be used. We also performed some tests with CoSIP in a P2P mode with OpenDHT [115] and it worked fine. Therefore, our implementation of CoSIP supports three different operation modes:

- DHT-only mode: here the CoSIP proxy makes `STORE`s and `GET`s out of `REGISTER` and `INVITE` methods coming from UAs. No SIP server is involved.

- Cooperative mode: this mode is the main idea behind CoSIP where both the DHT and the SIP server are involved.

- Server-only mode: here the CoSIP proxy forwards the SIP signaling between UAs and the SIP server. Except adding and removing a `VIA` header, no modification to the SIP messages or further processing is undertaken.

Figure 4.7 shows a screenshot of our CoSIP proxy implementation. The proxy is running in DHT-only mode. It has received a SIP `REGISTER` message from the SIP UA (Ekiga) and performed a successful registration in the DHT.

A.2 Message Processing

If a CoSIP Proxy receives a `REGISTER` from one of its UAs, it adds a VIA header to the message and forwards the modified message to the SIP Server. Furthermore, it stores the contact data of the user in the DHT.

```
STORE(H(alice@example.org), alice_IP:alice_port)
```

Two different timers *i)* for the server registration and *ii)* for the DHT `STORE` RPC are started. If the server does not respond in time, we assume that the server is unreachable for some reasons[2], and register the UA to the DHT only. Registration in the DHT fails in the unlikely case that the CoSIP proxy looses connectivity to the rest of the DHT. In case of success of either registration at the server or at the DHT, the CoSIP proxy responds with a `200 OK` SIP message to the UAC.

As we will see in Chapter 7, the CoSIP proxy may need to wait for a successful registration at the SIP server before the registration in the DHT is possible. This is the case if the CoSIP proxy needs to acquire a certificate for the UA upon successful registration. Using this certificate, the contact data stored in the DHT can be integrity-protected. However, in case the certificate is still valid, registration in the DHT can be initiated simultaneously with the registration at the server.

A.2.1 Session Establishment with CoSIP

If the CoSIP proxy receives an `INVITE` message from a UAC, it adds a `VIA` header to the message and forwards it to the SIP server. It sends a `100 Trying` message to the UAC back. Furthermore, the CoSIP proxy tries to resolve the SIP URI of the Callee, let's say `bob@example.org`, to the location of the Callee's CoSIP proxy using the DHT:

```
(bob_IP:bob_port) = GET(H(bob@example.org))
```

Two different timers are initiated to limit the response time of the SIP server and the DHT.

[2] Message loss is processed separately at the SIP transaction layer.

- If the server responds first, the response is forwarded to the UA. A subsequent response from the DHT is suppressed.
- If the DHT responds first, a subsequent response from the server is suppressed. The data received from the DHT is used to send the `INVITE` message directly to the Callee's CoSIP Proxy. The Callee's CoSIP Proxy forwards the `INVITE` message to the Callee. The response from the Callee (e.g., `200 OK`) traverses both CoSIP proxies back to the UAC.
- If neither server nor DHT respond in time, the CoSIP proxy sends an error message `408 Request Timeout` to the UAC.

A.3 High-Level State Machines

A CoSIP proxy keeps state information for each UA to process user registration and session establishment. Note however that the CoSIP proxy is not a full-fledged stateful SIP proxy according to RFC 3261 [120]. It rather keeps a minimal state to coordinate the parallel communication with the server and the DHT.

The state machine of the CoSIP proxy is organized in a modular and hierarchical approach. The CoSIP proxy can deal with one or more UACs behind it by logically separating their state machines. Furthermore, for each UA the state machine is separated according to the SIP sessions. We differentiate between `REGISTER` and `INVITE` session state machines. As in standard SIP, different sessions can be differentiated according to the `Call-ID`.

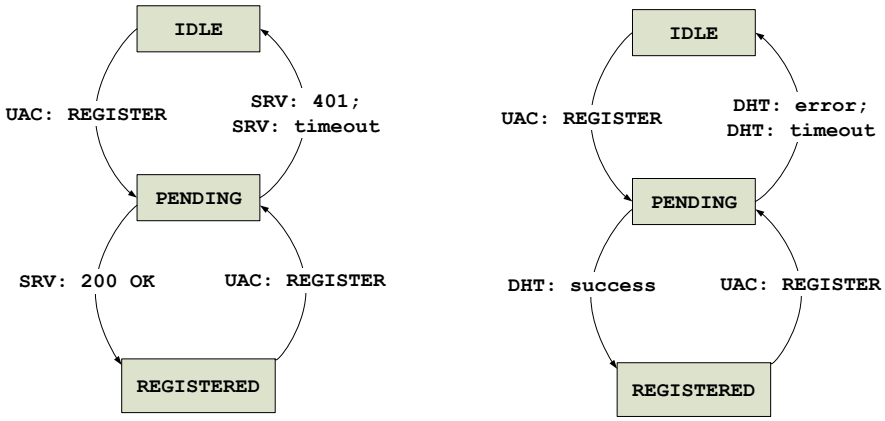
Moreover, each session state machine is separated according to the current state of the communication with *i)* the server and *ii)* the DHT. For example, the `REGISTER` session state machine shown in Figure A.1 is separated into two sub-state machines `SRV.*` and `DHT.*`. The state of the session is the compound state of both sub-state machines. For example, if the state is (`SRV.REGISTERED` $\wedge$ `DHT.REGISTERED`), this means that the registration has successfully been performed at both the server and the DHT. This results into a clean and unambiguous hierarchy:

UAC $\rightarrow$ session (`REGISTER` or `INVITE`) $\rightarrow$ sub-state machine (`SRV` or `DHT`)

For each event, notably an incoming message or a timeout, the CoSIP proxy identifies the sub-state machine uniquely and performs the required processing.

A.3.1 `REGISTER` Session State Machine

A `REGISTER` session is created, if a UA sends a `REGISTER` message via the CoSIP Proxy to the SIP server. The initial state of the `REGISTER` session is (`SRV.IDLE` $\wedge$ `DHT.IDLE`). The CoSIP proxy forwards the `REGISTER` message



(a) `SRV.*` sub-state machine (b) `DHT.*` sub-state machine

Fig. A.1: CoSIP `REGISTER` State Machine.

to the SIP server, starts a `SRV Timer` for limiting the server response time and switches the state of the `SRV` sub-state machine to `SRV.PENDING`.

If the server requires authentication, it responds with a `401 Unauthorized` message. In this case, the state is set back to `SRV.IDLE`. By the time the next `REGISTER` message is received (which should contain the appropriate authentication credentials), the state is switched back to `SRV.PENDING`. If a `200 OK` message is received from the server in state `SRV.PENDING`, the state is switched to `SRV.REGISTERED`. If the `SRV Timer` expires in state `SRV.PENDING`, we assume that the server is currently not reachable. The registration at the server fails.

As for the `DHT` sub-state machine, the CoSIP proxy performs a `STORE` request and moves to the state `DHT.PENDING`. If the DHT registration succeeds, the state is set to `DHT.REGISTERED`. In the unlikely case that registration at the DHT fails, the state is reset to `DHT.IDLE`.

The registration of a UA to a SIP server expires within a certain period. Thus, the UA needs to renew the registration periodically. If the CoSIP proxy receives a `REGISTER` message, a new registration cycle is started and the state of both sub-state machines is rest to `PENDING`.

A.3.2 `INVITE` Session State Machine

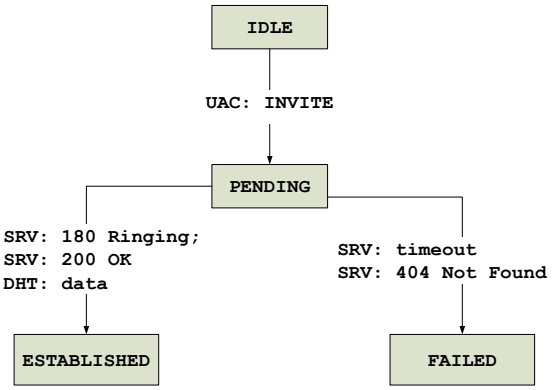
Upon receipt of an `INVITE` message from a UAC, the CoSIP proxy creates a new `INVITE` session with two sub-state machines `SRV` and `DHT`. Both sub-state machines start in the respective state `IDLE`. The CoSIP proxy switches the `SRV` sub-state to `PENDING` immediately after forwarding the `INVITE` message to the SIP server. A `SRV Timer` is started. The `SRV` sub-state is switched to

ESTABLISHED upon receipt of a 180 RINGING or 200 OK response from the server. and to FAILED upon a SRV Timeout event.

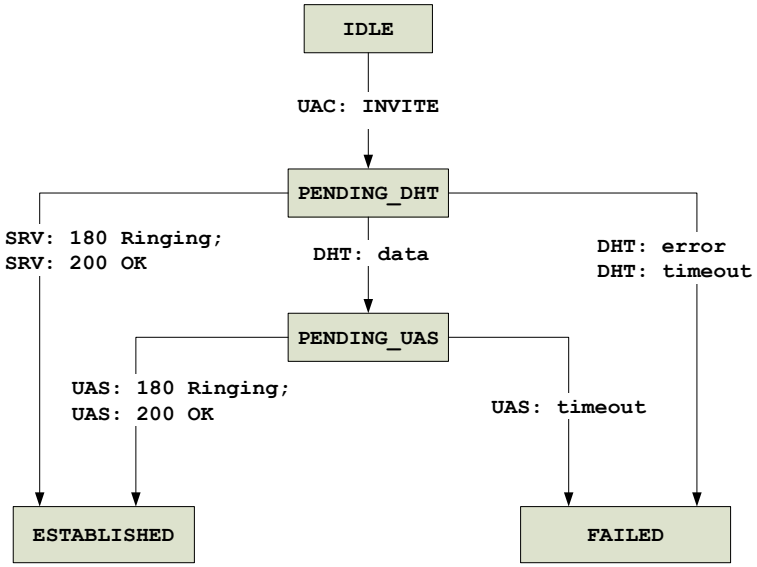
As for the DHT sub-state machine, the CoSIP proxy initiates the lookup in the DHT upon the initialization of the INVITE session and switches the DHT sub-state immediately to PENDING_DHT. If the lookup in the DHT fails, e.g., either because the Caller's CoSIP proxy is encountering connectivity problems to the DHT, or no data in the DHT is found, then the DHT sub-state is switched to FAILED. If the DHT lookup is successful, the CoSIP proxy sends the INVITE message to the Callee's CoSIP proxy and switches the DHT sub-state machine to PENDING_UAS.

In the state PENDING_UAS, upon receipt of a 180 RINGING or 200 OK response from the Callee's CoSIP proxy the sub-state is switched to ESTABLISHED. Upon a UAS Timeout event, the sub-state is switched to FAILED. The case of UAS Timeout occurs, e.g., if the Callee has gone offline and its contact data stored in the DHT is outdated.

Finally, in both sub-state machines, the state is moved to ESTABLISHED as soon as either the server or the DHT responds. This guarantees that if a response is received from the server, then any subsequent response from the DHT is suppressed and if a response is received from the DHT is received, then any subsequent response from the server is suppressed. The motivation behind this is to avoid race conditions in the signaling at the Caller UA and make the "forked" signaling at the CoSIP proxy fully transparent to the UA.



(a) SRV.* sub-state machine



(b) DHT.* sub-state machine

Fig. A.2: CoSIP INVITE State Machine.

B. SIP-BASED X.509 CERTIFICATE ENROLLMENT

An authentication protocol between a peer and a supervisor to acquire the appropriate certificates was presented in Section 7.3.1. In this chapter, we describe how SIP can used as transport "transport protocol" to enroll the certificates upon successful user registration. We implemented the authentication protocol as an extension of our CoSIP implementation (See Chapter A). In contrast to legacy SIP where a `REGISTER` message does not carry a message body, a certificate request is carried as message body. Upon successful authentication, the UA receives a X.509 certificate in the `200 OK` message. We extended the SER authentication module as well as our CoSIP proxy implementation for this purpose. Figure B.1 highlights the differences between a legacy SIP user registration message flow and CoSIP user registration message flow.

Listings B.1 and B.2 (Below at the end of this chapter) show the `REGISTER` message and the `200 OK` message from traces of a successful authentication protocol run using our implementation. The `REGISTER` message includes the Base64-encoded certificate request. The Authorization Header includes the challenge from the server (*nonce*) and the corresponding response which has been computed by the UA based on the preshared key. The `Content-Type` is `application/pkcs10`, i.e., a certificate request [148]. `Content-Length` and `Content-Transfer-Encoding` are additional headers required to process the message body which otherwise does not exist in legacy SIP registration. The `200 OK` response from the server which includes the X.509 certificate.

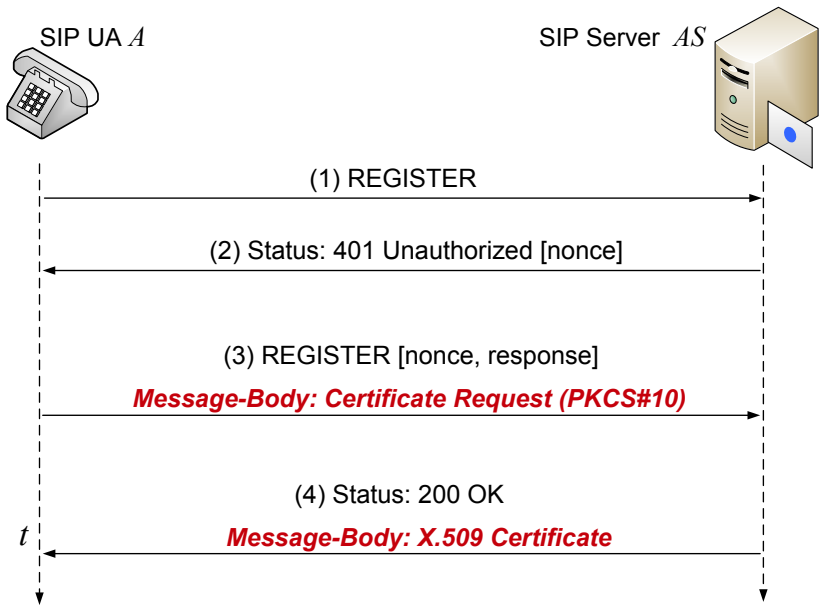


Fig. B.1: CoSIP registration

Listing B.1: CoSIP `REGISTER` message with a certificate request in the message body.

```
1  REGISTER sip:192.168.1.13 SIP/2.0
2  From: "alice" <sip:alice@192.168.1.13>
3  To: "alice" <sip:alice@192.168.1.13>
4  Via: SIP/2.0/UDP 192.168.1.13:10000;branch=z9hG4bK4D2753BD
5  Call-ID: 1795242434@192.168.1.13
6  CSeq: 7019 REGISTER
7  Contact: "alice" <sip:alice@192.168.1.13:10000;transport=udp>;
       methods="INVITE,MESSAGE,INFO,SUBSCRIBE,OPTIONS,BYE,CANCEL,
       NOTIFY,ACK,REFER"
8  Authorization: Digest username="alice", realm="192.168.1.13",
       nonce="4900d5c0000000001a93b10143b4d83179311d18f24f53c8", uri
       ="sip:192.168.1.13", cnonce="abcdefghi", nc=00000001,
       response="b00fae2e774203be0de6edc60ee93025", opaque="",
       algorithm="MD5"
9  Expires: 900
10 Content-Length: 525
11 Content-Type: application/pkcs10
12 Content-Transfer-Encoding: base64
13 User-Agent: kphone/4.2
14 Event: registration
15 Allow-Events: presence
16
17 -----BEGIN CERTIFICATE REQUEST-----
18 MIIBTDCBtgIBADANMQswCQYDVQQGEwJVUzCBnzANBgkqhkiG9w0BAQEFAAOBjQAw
19 gYkCgYEA4ygxLtsL3PRIAnblaBoD1MlEVCtuQfHB51rXngeCc6Kioyly9tg8DBXw
20 ZP0/z5NRu+SKyoC+9lkl2eGiSWZ27ve50I5VsKhLSzgg36UJ6KoheDJkilrxZ/2R
21 xmoVe6zx2R86VSRBYVat6dxpUjwUw4PgMW+qeVn9WvHUNd1FUk8CAwEAAaAAMA0G
22 CSqGSIb3DQEBBQUAA4GBAJ7VQ+xUb99U069mRXRonnIxO0236D3D/+2FhuXalkBo
23 LRKQLXMXeiAJjzjWvIQX9uT0F/7X4UO8xHTjlM8DFWyr8Re+YZ5oeDzi3hVAck4p
24 sDvP9Cu6ICNHrmUV93uUf9ed7o3Dk/wPKvNBqriKiQGDnLWFtJ677Qp2ikCNcUg/
25 -----END CERTIFICATE REQUEST-----
```

Listing B.2: CoSIP `200 OK` message with a X.509 certificate in the message body.

```
SIP/2.0 200 OK
From: "alice" <sip:alice@192.168.1.13>
To: "alice" <sip:alice@192.168.1.13>;tag=650
    b16d746387f5eaabb6f44b302fd3d.4fc2
Via: SIP/2.0/UDP 192.168.1.13:10000;branch=z9hG4bK4D2753BD
Call-ID: 1795242434@192.168.1.13
CSeq: 7019 REGISTER
Content-Type: application/pkix-cert
Content-Transfer-Encoding: base64
Server: Kamailio (1.4.1-notls (i386/linux))
Content-Length: 603

-----BEGIN CERTIFICATE-----
MIIBkDCB+gIJAPBYEyxmEWKiMA0GCSqGSIb3DQEBBQUAMA0xCzAJBgNVBAYTAlVT
MB4XDTA4MTAyMzE5NDYyOFoXDTA4MTEwMjE5NDYyOFowDTELMAkGA1UEBhMCVVMw
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMIufUsxPnVo/uUhe5smeiX2YZw8
s1dy1Ll8WW2cVPxM3gZ+Tz3O+5XLbRh3NrFkZs8IbAt3T7uur5BqsdAJA0AcMLuR
qRs9raJ2IrRaofhmpfP32Mo0doFwxabGq5sXswiUGtuUYoBl6IiBJEMSseohLaic
dyY5k5F1K5nBo/jvAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEALzyYltT/+15XpA7+
/86P9u10LyLi31DyZo/CeKFNW7fLivMoVMpOYaVF4TL7ybVPv4HQ/zAhCnBcxHgk
6W3yxRHsZUH7/WKOtwavvvRuu9Qf6dGbiZXtpTPLpC+0aLHNVlSfOV7Bn78Ybita
+QBonxxCINEDoPlepaWoo/cwApE=
-----END CERTIFICATE-----
```

C. TAXONOMY OF ATTACKS ON P2P NETWORKS

In this chapter, we provide an attack taxonomy to evaluate to what extend the security mechanisms described in Section 7 do successfully remediate attacks on P2P networks, notably in the context of CoSIP. The attack taxonomy is classified in a layered approach into:

i) Attacks on the overlay,

ii) Attacks on the DHT,

iii) Attacks on the application.

The impact of attacks on one of these layers may propagate to upper layers. For example, attacks targeting the overlay routing availability may lead to the failure of storing or retrieving content in or from the DHT. This in turn may lead to the failure of SIP user registration in the DHT, or SIP session establishment [1].

Some attacks are valid at several layers. For example, flooding attacks may be performed at the overlay layer by generating overlay signaling messages, or at the application layer by generating SIP signaling messages. We will mention attacks which are valid at different layers at each of the corresponding layers for completeness.

C.1 Attacks on the Overlay

Message Manipulation

An attacker may tamper with, insert or discard messages it is forwarding in the overlay.

Impact: Loss of overlay signaling integrity. Loss of overlay signaling availability if overlay messages are discarded.

Countermeasures:

[1] This is, in fact, similar to the propagation of the impact of attacks in the Internet layers. For example, attacks on a cable may lead to the unavailability of IP connectivity between two IP hops. A DoS attack on a IP router may lead to the lack of availability of web content

- Integrity protection of overlay signaling to detect forged messages.
- Parallel and iterative overlay routing to counter message discarding.

Request Hijacking

Request hijacking is a special case of message manipulation. An attacker intercepts a request from peer A and sends a forged response. Additional mechanisms may be required by the attacker to prevent the correct responder from sending a response or to prevent the correct response from reaching the request initiator.

Impact: Loss of overlay signaling integrity.

Countermeasures: Integrity protection of overlay signaling. Parallel routing to counter illegal message interception.

Peer Impersonation

A malicious node uses a fake ID in the P2P network or misuses the ID of another peer.

Impact: Loss of overlay signaling integrity.

Countermeasures: Cryptographically verifiable peer IDs. Integrity protection of overlay signaling.

Invalid Message Forwarding

A malicious node may forward overlay messages to an invalid node, non-existing nodes or existing but random nodes, or simply discard the overlay messages.

Impact: Higher overlay routing latency. Degradation of overlay routing availability.

Countermeasures: Parallel and iterative overlay routing.

Propagating Wrong Routing Tables

A malicious node may even propagate wrong routing information and force other honest nodes to forward overlay messages incorrectly.

Impact: Incorrect routing tables result into high overlay routing latency and degradation of overlay routing availability.

Countermeasures:

- Routing table entries should be validated before they are adopted. This can be performed by the following mechanisms:
  - Each routing entry propagated should include the node ID, location (IP and port) and certificate. This prevents attacks where malicious nodes propagate routing entries with non-existing nodes.
  - Additionally, a node must reply to an overlay layer `Ping` message before it is adopted in the routing table. This prevents attacks where malicious nodes propagate routing entries with existing but offline nodes.
- Rejection of faulty or stale routing tables entries.
- Parallel and iterative overlay routing to efficiently route beyond faulty or stale routing table entries.

Bootstrap Attacks

If a honest node contacts a malicious peer to join the P2P network, the malicious peer may provide the honest peer with wrong overlay information.

Impact: The honest node will join a parallel overlay network ⇒ Loss of availability.

Countermeasures: Node certificates must include an identifier of the overlay, such as new nodes can verify that they are in the correct overlay. For example, in X.509 certificates the node ID can be in the Subject Common Name (CN) and the overlay name in the Organization (O) or Organizational Unit (OU).

Chosen-Location Attacks

The attacker chooses an ID in the overlay such as it located in a strategically good location in the overlay, e.g., in the vicinity of their targeted victim peer. A chosen-location attack does not cause harm by itself. It rather increases the success probability of subsequent attacks.

Impact: By choosing a location in the vicinity of a victim A, the attacker M increases the probability to be responsible for forwarding messages from or to A. This may lead to loss of overly routing integrity or availability.

Countermeasures: A peer must not be able to choose its own ID or even choose a preferred sector of the ID space. The overlay supervisor is responsible for generating node IDs.

Flooding Attacks

An attacker may launch a DoS attack against one or more nodes in the overlay by flooding them with a large number of overlay messages, e.g., routing requests.

Impact: Increased CPU, memory and bandwidth usage at the victim peer ⇒ Resource depletion. Degradation of overlay availability.

Countermeasures: Typical mechanisms against flooding attacks. For example, cookies [77] or puzzles [7] which can be generated by the victim and verified with low CPU resources. They may help against attackers with spoofed IP addresses. However, network congestion may not be prohibited successfully. The benefits of cookies and puzzles in case of large scale distributed DoS attacks are limited.

DoS Attack Amplification by Misusing Recursive Routing

Using recursive routing; each lookup message in the network generates $O(log\, n)$ messages. This amplifies flooding attacks.

Impact: Increased CPU, memory and bandwidth usage in the P2P network ⇒ Resource depletion. Degradation of availability.

Countermeasures: Iterative routing ⇒ Each message generated by an attacker generates a single response.

DoS Attack Amplification based on Churn Enforcement

Some DHT algorithms, e.g., Chord [144] and Pastry [123] require adapting overlay routing tables each time a new node joins or leaves the network. Malicious nodes may misuse this to generate a heavy load in the network by joining and leaving the network frequently. This can be misused to amplify DoS attacks.

Impact: Increased CPU and bandwidth usage in the P2P network ⇒ Resource depletion. Degradation of availability.

Countermeasures: Dampening the impact of joins and leaves. New joining nodes should not cause immediate major changes in the routing tables of other peers.

Sybil Attacks

Sybil attacks [37] are a generalization of peer impersonation attacks. A malicious node joins the P2P network with multiple fake identities.

Impact: Other nodes believe that they are interacting with different nodes, while they are actually interacting with the same node. A malicious node performing a Sybil attack may be able to gain control over a large part of the P2P network. A Sybil attack does not cause harm by itself. It rather increases the success probability of subsequent attacks.

A Sybil attack increases the probability that an attacker disposes of a message sent from A to B and thus increases the ability to manipulate it.

Countermeasures: Verifiable peer IDs signed by a central authority. The central authority needs to control the number of assigned verifiable peer IDs per user.

Node Eclipse Attacks

Node eclipse attacks [135] are a special case of message and data manipulation attacks. They are called after the term 'eclipse' in astronomy which means that an object moves into the shadow of another. In the context of P2P networks, a node eclipse attack is an attack where a node is shielded by an attacker.

Impact: A node eclipse attack on a node renders all communication from and to that node controlled by the attacker. This leads to loss of integrity and availability of the overlay signaling.

The basis for a successful eclipse attack may be a combination of a Sybil attack and chosen-location attacks.

Countermeasures:

- Prevent Sybil and chosen-location attacks.

- Parallel and iterative overlay routing to increase the probability to route beyond the eclipse attacker.

Inconsistent Behavior

Malicious peers may behave inconsistently in order to not reveal their actual intention and depending on the situation. Any of the attacks on the overlay above may be combined with inconsistent behavior. For example, they may occasionally discard overlay messages they are supposed to forward. A malicious node M may always respond to keepalive messages from a honest peer A in order to convince A to keep M in her overlay routing table. However, M may still discard incoming overlay messages which should be forwarded to A.

Impact: Higher overlay routing latency and potentially lack of overlay routing availability.

Countermeasures: Parallel and iterative overlay routing to increase the probability to route beyond the inconsistent attacker.

C.2 Attacks on the DHT

Content Manipulation

An attacker may tamper with, insert or delete data items in the DHT. A malicious peer may delete a data item locally or may send a `DELETE` request to other peers storing the data item.

Impact: Loss of DHT content integrity or DHT content availability.

Countermeasures: Integrity protection of data stored in the DHT. Data replication to alleviate the impact of data discarding. DHT content must be protected from illegitimate delete operations by verifying that the peer sending the `DELETE` request is the verifiable "owner" of the data item.

Chosen-Location Attacks

Chosen-location attacks may target a victim node (See Section C.1 above) or DHT content, e.g., the contact data of a SIP UA.

Impact: By choosing a location in the vicinity of the key for a data item m, the attacker M increases the probability to be responsible for m. This

allows for DHT content manipulation. This may lead to loss of DHT content routing integrity or availability.

Countermeasures: Same as countermeasures for chosen-location attacks at the overlay layer (See Section C.1).

DoS Attack Amplification based on Churn Enforcement

Churn may cause not only immediate changes in the routing tables as mentioned in Section C.1. For example, Chord [144] and Pastry [123] require also reshuffling data each time a new node joins or leaves the network. This can be misused to amplify DoS attacks.

Impact: Increased CPU and bandwidth usage in the P2P network ⇒ Resource depletion. Degradation of availability.

Countermeasures: Dampening the impact of joins and leaves. Stored data should not be reshuffled. Instead, the data publisher should be responsible for refreshing the data by himself and guaranteeing that the right replica nodes dispose of it.

Sybil Attacks

Sybil attacks may target the DHT content as well.

Impact: A Sybil attack increases the probability that the attacker will be responsible for storing a data item. This allows for the manipulation of that data item. Furthermore, the attacker will be able to control to whom it wants to provide the data item. This leads to lack of DHT content integrity and availability.

Countermeasures: Same as Sybil attacks on the overlay. See Section C.1.

Content Eclipse Attacks

A content eclipse attack is an attack where a data item in the DHT is shielded by an attacker.

Impact: A content eclipse attacks renders the data item unavailable for lookup from other peers which, i.e., loss of content availability.

Countermeasures:

- Prevent Sybil and chosen-location attacks.
- Parallel and iterative lookup to increase the probability to reach one of the replica nodes.
- Sufficient data replication by the publisher to reduce the success probability of a content eclipse attack.

Inconsistent Behavior

Malicious peers may behave inconsistently in DHT as well. For example, they may discard data they are supposed to store while still responding positively to a `STORE` RPC.

Impact: Lack of DHT content availability.

Countermeasures: Sufficient replication.

C.3 Attacks on the Application

UA Impersonation

A malicious UA uses a fake SIP URI or steals the URI of another UA.

Impact: Loss of SIP signaling integrity.

Countermeasures: Verifiable SIP URIs. Integrity protection of SIP signaling.

Spam over IP Telephony (SPIT)

Impact: User annoyance.

Countermeasures: Accept phone calls only from known UAs. Obviously this countermeasure reduces the application availability though.

SIP Flooding Attacks

SIP flooding attacks have the same properties in terms of impact and countermeasures as the flooding attacks at the overlay layer (See Section C.1).

Eavesdropping & Traffic Analysis

A malicious node may passively record activities of other users or peers in the network. A malicious node may use this information to build profiles with locations and social interaction of its neighbors, e.g., when and whom they are calling.

Impact: Loss of confidentiality. Loss of privacy.

Countermeasures: Encryption of SIP and overlay signaling. Deployment of anonymization techniques (See Chapter 8).

BIBLIOGRAPHY

[1] Overlay network, Network overlays. http://www.overlay-networks.info/.

[2] The Gnutella 0.4 protocol specification, 2000. http://dss.clip2.com/GnutellaProtocol04.pdf.

[3] 3GPP. IP Multimedia Subsystem (IMS); Stage 2. TS 23.228, 3rd Generation Partnership Project (3GPP), Sept. 2008.

[4] J. Abley and K. Lindqvist. Operation of Anycast Services. IETF RFC 4786 (Best Current Practice), Dec. 2006.

[5] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.

[6] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein. A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability. *IEEE Communications Surveys & Tutorials*, 11(2), 2009.

[7] T. Aura, P. Nikander, and J. Leiwo. DOS-Resistant Authentication with Client Puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 170–177, London, UK, 2001. Springer-Verlag.

[8] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[9] A. Back, U. Möller, and A. Stiglic. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In I. S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[10] S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *INFOCOM*. IEEE, 2006.

[11] I. Baumgart and S. Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *ICPADS '07: Proceedings of the 13th International Conference on Parallel and Distributed Systems*, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[12] S. Becker, R. State, and T. Engel. Using game theory to configure P2P SIP. In *Proceedings of IPTComm '09, Atlanta, Georgia*, pages 1–9. ACM, 2009.

[13] A. Beier. VoIP-Störung bei United Internet, July 2006. http://www.heise.de/newsticker/meldung/75382.

[14] CIDR Report, 2010. `http://www.cidr-report.org/as2.0/`.

[15] L. N. Bhuyan and D. P. Agrawal. Generalized Hypercube and Hyperbus Structures for a Computer Network. *IEEE Trans. Comput.*, 33(4):323–333, 1984.

[16] A. Binzenhöfer and H. Schnabel. Improving the Performance and Robustness of Kademlia-based Overlay Networks. In *KIVS 2007*, Bern, Switzerland, Feb. 2007.

[17] H. Bleich. Angriff auf DNS-Server von InternetX, July 2010. `http://www.heise.de/newsticker/meldung/Angriff-auf-DNS-Server-von-InternetX-897712.html`.

[18] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 92–102, New York, NY, USA, October 2007. ACM.

[19] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 1 edition, September 2003.

[20] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[21] D. A. Bryan. P2PSIP: On The Road To A World Without Servers. *BUSINESS COMMUNICATIONS*, pages 40–44, April 2007.

[22] D. A. Bryan, B. B. Lowekamp, and C. Jennings. SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System. In *AAA-IDEA '05: Proceedings of the First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, pages 42–49, Washington, DC, USA, 2005. IEEE Computer Society.

[23] J. Byers, J. Considine, and M. Mitzenmacher. Simple Load Balancing for Distributed Hash Tables. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, pages 80–87, 2003.

[24] IP als Kommunikationsbasis, 2010. `http://www.eenova.de/projekte/seis/kommunikationsbasis/`.

[25] G. Camarillo. *SIP Demystified*. McGraw-Hill Professional, 2001. Foreword By-Rosenberg, Jonathan.

[26] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 85–98, Berkeley, CA, USA, 2005. USENIX Association.

[27] S. Chen, X. Wang, and S. Jajodia. On the anonymity and traceability of peer-to-peer VoIP calls. *IEEE Network*, 20(5):32–37, 2006.

[28] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS Using a Peer-to-Peer Lookup Service. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 155–165, London, UK, 2002. Springer-Verlag.

[29] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a Common API for Structured Peer-to-Peer Overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA, February 2003.

[30] G. Danezis. Statistical Disclosure Attacks: Traffic Confirmation in Open Environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[31] G. Danezis, C. Lesniewski-laas, L. Tong, and L. Tong. Sybil-resistant DHT routing. In *In ESORICS*, page 305–318. Springer, Springer, 2005.

[32] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, 2007.

[33] Z. Despotovic and K. Aberer. P2P reputation management: probabilistic estimation vs. social networks. *Comput. Netw.*, 50(4):485–500, 2006.

[34] J. Dinger and H. Hartenstein. Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration, April 200.

[35] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[36] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. In *SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

[37] J. R. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[38] R. Ellison, R. J. Ellison, D. Fisher, D. A. Fisher, R. C. Linger, R. C. Linger, H. F. Lipson, H. F. Lipson, T. Longstaff, T. Longstaff, N. Mead, and N. R. Mead. Survivable Network Systems: An Emerging Discipline. Technical report, Carnegie Mellon University, 1997.

[39] I. P. M. et. al. I2P Tech Intro. `http://www.i2p2.de/techintro.html`.

[40] H. Federrath. JAP: Anonymity and Privacy, 2000-2006. `http://anon.inf.tu-dresden.de`.

[41] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. Technical report, Counterpane Internet Security, Inc, 2000.

[42] N. Ferguson and B. Schneier. *Practical Cryptography.* John Wiley and Sons (1st edition), 2003.

[43] M. J. Freedman, E. Sit, J. Cates, and R. Morris. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 121–129, London, UK, 2002. Springer-Verlag.

[44] Geo IP Tool - View my IP information. `http://www.geoiptool.com/`.

[45] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[46] M. Goyal, K. K. Ramakrishnan, and W.-C. Feng. Achieving faster failure detection in OSPF networks. In *IEEE International Conference on Communications (ICC)*, volume 1, 2003.

[47] J. L. Gross and J. Yellen. *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2005.

[48] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS'06: The 5th International Workshop on Peer-to-Peer Systems*, 2006.

[49] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 381–394, New York, NY, USA, 2003. ACM.

[50] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: a high performance, server-centric network architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*, 39(4):63–74, 2009.

[51] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03*. Springer-Verlag, 2003.

[52] A. Haider and R. Harris. Recovery Techniques in Next Generation Networks. *IEEE Communications Surveys*, 9(3), 2007.

[53] M. Handley. Why the Internet only just works. *BT Technology Journal*, 24, July 2006.

[54] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. IETF RFC 4566 (Proposed Standard), July 2006.

[55] 3GPP Specification detail; All-IP network (AIPN) feasibility study, September 2005. `http://www.3gpp.org/ftp/Specs/html-info/22978.htm`.

[56] C. Inc. X-Lite. `http://www.counterpath.com/x-lite.html`.

[57] D. Inc. Ekiga Free Your Speech. `http://ekiga.org/`.

[58] S. Ioannidis, G. Apostolopoulos, K. Anagnostakis, N. Nikiforakis, A. Makridakis, and C. Gkikas. Resilience of communication networks: Resilience features of IPv6, DNSSec and MPLS. Technical report, European Network and Information Security Agency (ENISA), 2009.

[59] P. Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall, April 1994.

[60] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. A SIP Usage for RELOAD. draft-ietf-p2psip-sip-04, IETF Internet Draft, work in progress, March 2010. `http://tools.ietf.org/html//draft-ietf-p2psip-sip-04`.

[61] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. REsource LOcation And Discovery (RELOAD). draft-ietf-p2psip-base-08, IETF Internet Draft, work in progress, March 2010. `http://tools.ietf.org/html/draft-ietf-p2psip-base-08`.

[62] A. Johnston. *SIP: Understanding the Session Initiation Protocol, Second Edition*. Artech House, Inc., Norwood, MA, USA, 2003.

[63] Jürgen Küri. Panne im T-Mobile-Netz wird untersucht [Update], April 2009. `http://www.heise.de/newsticker/meldung/Panne-im-T-Mobile-Netz-wird-untersucht-Update-214640.html`.

[64] H. J. Kang, E. Chan-Tin, N. Hopper, and Y. Kim. Why Kad Lookup Fails. In *Peer-to-Peer Computing*, pages 121–130, 2009.

[65] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663, New York, NY, USA, 1997. ACM.

[66] D. R. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 36–43, New York, NY, USA, 2004. ACM.

[67] A. R. Karthik, A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load Balancing in Structured P2P Systems. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, pages 68–79. Springer-Verlag, 2003.

[68] C. C. Keir, C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, pages 273–286, 2005.

[69] A. S. Khan. Vietnam's submarine cable 'lost' and 'found', June 2007. `http://lirneasia.net/2007/06/vietnams-submarine-cable-lost-and-found/`.

[70] Y. Kitamura, Y. Lee, R. Sakiyama, and K. Okamua. Experience with Restoration of Asia Pacific Network Failures from Taiwan Earthquake. *IEICE Transactions on communications*, Volume E90, No. 11, 2007.

[71] L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delay*. McGraw-Hill, 1964.

[72] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, New York, NY, USA, 2007. ACM.

[73] J.-C. Laprie. Dependable Computing and Fault Tolerance: Concepts and Terminology. In *Proceedings of the 15th IEEE Int. Symp. Fault Tolerant Computing (FTCS-15)*, Ann Arbor, MI, June 1985.

[74] J.-C. Laprie. From Dependability to Resilience. In *38th International Conference On Dependable Systems and Networks*. IEEE/IFIP, 2008.

[75] J. C. Laprie, A. Avizienis, and H. Kopetz, editors. *Dependability: Basic Concepts and Terminology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.

[76] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the Kazaa Network. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 112, Washington, DC, USA, 2003. IEEE Computer Society.

[77] J. Lemon. Resisting SYN Flooding DoS Attacks with a SYN Cache. In *USENIX BSDCon'2002*, 2002.

[78] B. Leong and J. Li. Achieving One-Hop DHT Lookup and Strong Stabilization by Passing Tokens. In *Proceedings of the 12th International Conference on Networks 2004 (ICON 2004), Singapore, November 2004.*, 2004.

[79] B. Leong, B. Liskov, and E. D. Demaine. EpiChord: Parallelizing the Chord lookup algorithm with reactive routing state management. *Comput. Commun.*, 29(9):1243–1259, 2006.

[80] C. Lesniewski-Laas. A Sybil-proof one-hop DHT. In *SocialNets '08: Proceedings of the 1st Workshop on Social Network Systems*, pages 19–24, New York, NY, USA, 2008. ACM.

[81] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 395–406, New York, NY, USA, 2003. ACM.

[82] U. Mansmann. VoIP-Ausfall bei 1&1, January 2010. `http://www.heise.de/newsticker/meldung/VoIP-Ausfall-bei-1-1-Update-895875.html`.

[83] S. Marti and H. Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. Working Paper 2005-11, Stanford InfoLab, 2005.

[84] D. Massey. A Comparative Study of the DNS Design with DHT-Based Alternatives. In *In the Proceedings of IEEE INFOCOM'06*, 2006.

[85] MaxMind Inc. Geolocation and Online Fraud Prevention from MaxMind. `http://www.maxmind.com/`.

[86] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *Peer-To-Peer Systems: First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002*, pages 53–65, 2002.

[87] J. McLachlan and N. Hopper. Don't Clog the Queue! Circuit Clogging and Mitigation in P2P Anonymity Schemes. In *Financial Cryptography*, pages 31–46, 2008.

[88] M. Menth, R. Martin, A. Koster, and S. Orlowski. Overview of Resilience Mechanisms Based on Multipath Structures. In *Proceedings of the 6th International Workshop on Design of Reliable Communication Networks (DRCN)*, 2007.

[89] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. AP3: cooperative, decentralized anonymous communication. In *EW 11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 30, New York, NY, USA, 2004. ACM.

[90] L. Monnerat, , L. R. Monnerat, and S. E. Ti/ti-e. D1HT: A Distributed One Hop Hash Table. Technical report, In Proc of the 20th IEEE Intl Parallel and Distributed Processing Symposium (IPDPS, 2005.

[91] E. F. Moore and C. E. Shannon. Reliable circuits using less reliable relays, Part I: Introduction, Part II: The central problem. *Journal of the Franklin Institute*, 262:191–208, 281–297, September 1956.

[92] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, Washington, DC, USA, May 2005. IEEE Computer Society.

[93] A. Nambiar and M. Wright. Salsa: A Structured Approach to Large-Scale Anonymity. In *Proceedings of CCS 2006*, October 2006.

[94] S. Niccolini. SPIT Prevention: State of the Art and Research Challenges. In *the 3rd VoIP Security Workshop*, Berlin, 2006.

[95] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 39–50, New York, NY, USA, 2009. ACM.

[96] K. Ono and H. Schulzrinne. Principles, systems and applications of ip telecommunications. services and security for next generation networks. chapter One Server Per City: Using TCP for Very Large SIP Servers, pages 133–148. Springer-Verlag, Berlin, Heidelberg, 2008.

[97] L. Øverlier and P. Syverson. Locating Hidden Servers. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 100–114, Washington, DC, USA, May 2006. IEEE Computer Society.

[98] IETF P2PSIP working group charter web page. `http://www.ietf.org/dyn/wg/charter/p2psip-charter.html`.

[99] K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS: improving DNS performance and reliability via cooperative lookups. In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, pages 14–14, Berkeley, CA, USA, 2004. USENIX Association.

[100] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470, 2007.

[101] B. Pfitzmann and M. Waidner. Federated Identity-Management Protocols. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Proceedings of 11th International Workshop on Security Protocols*, volume 3364/2005, pages 153–174. Springer-Verlag, LNCS, April 2003.

[102] Open Source Portable SIP Stack and Media Stack for Windows and Mac OS X, 2010. `http://www.pjsip.org/`.

[103] A. Popescu, B. J. Premore, and E. Zmijewski. Recent Cable Breaks - A Middle East & Persian Gulf Perspective. Presentation, Renesys Corp. Salmiya, Kuwait, April 2008.

[104] Princeton University. PlanetLab - An open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`.

[105] World Internet Usage Statistics News and World Population Stats, 2010. `http://www.internetworldstats.com/stats.htm`.

[106] QuteCom, Free VoIP Softphone. `http://www.qutecom.org/`.

[107] M. Raab and A. Steger. "Balls into Bins" - A Simple and Tight Analysis. In *RANDOM '98: Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer-Verlag, 1998.

[108] S. Rai, B. Mukherjee, and O. Deshpande. IP resilience within an autonomous system: current approaches, challenges, and future directions. *Communications Magazine, IEEE*, 43(10), Oct 2005.

[109] V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the Internet. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 331–342, New York, NY, USA, 2004. ACM.

[110] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[111] M. Rausand and A. Hoyland. *System Reliability Theory; Models, Statistical Methods, and Applications*. Addison-Wesley Publishing Company (2nd Edition), Reading, Massachusetts, 2004.

[112] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[113] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 91–102, New York, NY, USA, November 2002. ACM.

[114] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[115] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: a public DHT service and its uses. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 73–84, New York, NY, USA, 2005. ACM.

[116] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, Understanding, and Analyzing Critical Infrastructure Interdependencies. *IEEE control systems magazine.*, 21:11–25, 2001.

[117] RIPE NCC. YouTube Hijacking: A RIPE NCC RIS case study. `http://www.ripe.net/news/study-youtube-hijacking.html`.

[118] J. Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP). IETF RFC 3856 (Proposed Standard), Aug. 2004.

[119] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. IETF RFC 5245 (Proposed Standard), Apr. 2010.

[120] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. IETF RFC 3261, June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630.

[121] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). IETF RFC 3489 (Proposed Standard), Mar. 2003. Obsoleted by RFC 5389.

[122] M. Rosing. *Implementing elliptic curve cryptography.* Manning Publications Co., Greenwich, CT, USA, 1999.

[123] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Lecture Notes in Computer Science*, pages 329–350, 2001.

[124] A. I. T. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In *Symposium on Operating Systems Principles*, pages 188–201, 2001.

[125] F. R. Schreiber. *Sybil.* Warner Books, 1995.

[126] H. Schulzrinne. Engineering Peer-to-Peer Systems, September 2008. Key note at the 8th IEEE International Conference on Peer-to-Peer Computing (P2P'08) `http://www.p2p08.org/program/sessions/1-invited-talk-i/P2P08-keynote.pdf/at_download/file`.

[127] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550 (Standard), July 2003. Updated by RFC 5506.

[128] J. Seedorf. Security challenges for peer-to-peer SIP. *IEEE Network*, 20(5):38–45, 2006.

[129] J. Seedorf. Using Cryptographically Generated SIP-URIs to Protect the Integrity of Content in P2P-SIP. In *the 3rd VoIP Security Workshop*, Berlin, 2006.

[130] J. Seedorf. Lawful Interception in P2P-Based VoIP Systems. In *Proceedings of IPTComm 2008, Heidelberg, Germany, July 1-2, 2008*, pages 217–235, Berlin, Heidelberg, 2008. Springer-Verlag.

[131] A. Serjantov and P. Sewell. Passive Attack Analysis for Connection-Based Anonymity Systems. In *Proceedings of ESORICS 2003*, October 2003.

[132] K. Singh and H. Schulzrinne. Peer-to-Peer Internet Telephony using SIP. Technical report, Columbia University CUCS-044-04, 2004.

[133] K. N. Singh. *Reliable, Scalable and Interoperable Internet Telephony.* PhD thesis, Columbia University, New York, NY, 206.

[134] About SIP Express Router. `http://www.iptel.org/ser`.

[135] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269, London, UK, 2002. Springer-Verlag.

[136] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269, London, UK, 2002. Springer-Verlag.

[137] Restarts Cited in Skype Failure, August 2007. `http://www.nytimes.com/2007/08/21/business/worldbusiness/21skype.html`.

[138] Survivable Network Design Library, 2010. `http://sndlib.zib.de`.

[139] Sourceforge. KPhone. `http://sourceforge.net/projects/kphone/`.

[140] M. Steiner, T. En Najjary, and E. W. Biersack. A global view of KAD. In *IMC 2007, ACM SIGCOMM Internet Measurement Conference, October 23-26, 2007, San Diego, USA*, 10 2007.

[141] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith. Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines. *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, 2010. (to appear).

[142] J. P. G. Sterbenz, D. Hutchison, et al. ResiliNets Wiki. `https://wiki.ittc.ku.edu/resilinets_wiki/index.php/`.

[143] Steven Pigeon. Pairing Function. `http://mathworld.wolfram.com/PairingFunction.html`.

[144] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, New York, NY, USA, 2001.

[145] C. Tang, M. J. Buco, R. N. Chang, S. Dwarkadas, L. Z. Luan, E. So, and C. Ward. Low traffic overlay networks with large routing tables. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 14–25, New York, NY, USA, 2005. ACM.

[146] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1), 2004.

[147] Am Abend waren die Telekom-Leitungen tot, October 2007. `http://www.welt.de/wirtschaft/webwelt/article1313712/Am_Abend_waren_die_Telekom_Leitungen_tot.html`.

[148] S. Turner. The application/pkcs10 Media Type. IETF RFC 5967 (Informational), Aug. 2010.

[149] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. The Morgan Kaufmann Series in Networking, 2004.

[150] H. Wang, Y. Zhu, and Y. Hu. An Efficient and Secure Peer-to-Peer Overlay Network. In *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, pages 764–771, Washington, DC, USA, 2005. IEEE Computer Society.

[151] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 81–91, New York, NY, USA, 2005. ACM.

[152] A. Wilkens. Kundenanfragen überlasteten VoIP bei 1&1, November 2009. `http://www.heise.de/newsticker/meldung/Kundenanfragen-ueberlasteten-VoIP-bei-1-1-Update-858101.html`.

[153] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: provider portal for applications. *SIGCOMM Comput. Commun. Rev.*, 38(4):351–362, 2008.

[154] L. Zhuang, F. Zhou, U. C. Berkeley, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings of NSDI*. ACM/USENIX, 2005.

[155] S. Zoels, Z. Despotovic, and W. Kellerer. Cost-Based Analysis of Hierarchical DHT Design. In *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, pages 233–239, Washington, DC, USA, 2006. IEEE Computer Society.

ISBN 3-937201-17-3

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)