Analyzing and Mitigating Security Threats in P2P Systems

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieur (Dr.-Ing.)
vorgelegt von

M.Sc. Hatem Ismail

aus Kairo, Ägypten

Referenten:
Prof. Neeraj Suri, Ph.D.
Prof. Dr. Abdelmajid Khelil

Datum der Einreichung: 2. Juli 2018
Datum der mündlichen Prüfung: 13. September 2018

Darmstadt 2018
D17

Erklärung zur Dissertation

Hiermit versichere ich, *Hatem Ismail*, die vorliegende Dissertation ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 8th August 2018

_____

Abstract

Peer-to-Peer (P2P) protocols increasingly underlie a growing diversity of networked applications (e.g., file sharing, streaming multimedia, storage, VoIP) especially as the decentralized P2P paradigm inherently fosters scalability and robustness. The growing application-oriented services also result in the evolution of P2P systems spanning diverse data dissemination techniques, peer roles and topological structures.

On the flip side, while decentralization and scalability are attractive, and common for all P2P systems, these design features also increase the P2P network's exposure to a variety of security threats that can result in the degradation of services. In this thesis, we illustrate a set of important P2P attack types and subsequently develop approaches to *secure P2P networks* from these progressive and evolving attacks.

Covering a comprehensive progression of P2P systems of increasing complexity (i.e., structured, unstructured and streaming), we evaluate the corresponding feasibility of conducting attacks and the resultant impact onto them.

Subsequently, we investigate the progressive steps of detection, mitigation and sanitization potential to restore the requisite P2P functionality. Depending on the targeted P2P network model, we propose countermeasures that (a) are effective against a specific attack type and its possible variants, (b) are lightweight in execution, (c) are fully decentralized, i.e., do not depend on central entities, and (d) allow for both reactive and proactive mitigation.

Our theoretical analysis and simulations demonstrate that our proposed attack detection/mitigation mechanisms can reach up to 90-100% detection accuracy while inducing low overhead of 5-10% even when operating under severe attack scenarios.

Kurzfassung

Peer-to-Peer (P2P) Netzwerke unterliegen zunehmend einer sich steigernden Diversität vernetzter Anwendungen (z.B. File Sharing, Streaming, Multimedia, Datenspeicherung, VoIP), insbesondere infolge der durch das dezentralisierte P2P-Paradigma begünstigten Skalierbarkeit und Robustheit. Die zunehmende Zahl anwendungsorientierter Dienste führt auch zu einer Evolution von P2P-Modellen, die verschiedenartige Techniken der Datenverbreitung, sowie verschiedene Rollen der Peers und topologische Strukturen, umfasst.

Dezentralisierung und Skalierbarkeit sind attraktive Eigenschaften von allen P2P-Systemen, führen aber auch zu einer Anfälligkeit dieser Systeme gegenüber verschiedener Sicherheitsschwachstellen, die wiederum zu einer Minderung der Dienstgüte führen können. Die vorliegende Arbeit zeigt einen Satz wichtiger P2P-Angriffsarten auf und entwickelt Ansätze zum Schutz vor diesen sich abzeichnenden progressiven Angriffen.

Wir evaluieren die Durchführbarkeit und Auswirkungen dieser Angriffe für P2P-Modelle steigender Komplexität (Structured, Unstructured und Streaming).

Anschließend untersuchen wir aufeinander aufbauend Schritte der Angriffsdetektion und der Wirkungsminderung zur Wiederherstellung grundlegender P2P-Funktionalität. In Abhängigkeit vom betrachteten P2P-Modell schlagen wir Gegenmaßnahmen vor, die (a) wirksam gegen eine bestimmte Angriffsart einschließlich möglicher Varianten sind, (b) leichtgewichtig in der Ausführung sind, (c) vollständig dezentralisiert, also unabhängig von zentralen Entitäten, sind und (d) reaktive und proaktive Ansätze der Wirkungsminderung ermöglichen.

Unsere theoretische Analyse und Simulationen zeigen, dass die vorgestellten Mechanismen zur Detektion und Wirkungsminderung eine Detektionsgenauigkeit von 90-100% erzielen können und dabei selbst in massiven Angriffsszenarien einen niedrigen Mehraufwand von 5-10% erfordern.

Acknowledgments

Through years filled with strong and weak moments, climbing up and sometimes falling down, crawling in light and running in dark, that's how hard, enjoyable, stressful and successful my PhD journey was.

First, I am gratefully in debt to Prof. Neeraj Suri, my professor, supervisor and mentor through my PhD years, whom without, I would have never make it to writing those words now. I want to deeply thank him for teaching, supporting and continuously mentoring and pushing me forward and foremost, to give me the opportunity to be one of the wonderful DEEDS group.

Starting with my parents, for whom I can't even start explaining their unconditional love and support. Without them, I would have never been thinking about achieving this dream. For my colleagues, my other family here, I can't help but remember our serious and fun times, all the technical discussions and deadlines we shared. I really thank everyone of them for being a true friend and a very skilled and talented colleague from whom I learned a lot and enlightened my scopes.

Habib, Olli, Nicolas, Stefan, Tsveti, Heng, Yiqun, Salman, Patrick, Ute and Sabine, I want to thank them all for always being there and always make things easier through their dedication, friendliness, persistence and joyfulness. We will always be friends. Last but not least, special thanks to Daniel Germanus and Stefanie Roos, my colleagues and my mentors for all the hard work and for teaching me a lot through our journey with the different papers we have worked on together.

Contents

Erklärung zur Dissertation **i**

Abstract **iii**

Kurzfassung **v**

Acknowledgments **vii**

1 Introduction **1**
1.1 P2P Overlays . . . 2
1.2 Attacks on P2P Overlays . . . 4
1.3 Research Questions and Contributions . . . 6
1.4 Thesis Structure . . . 7
1.5 Publications . . . 8

2 Structured: LEA-Detection **9**
2.1 Problem Statement . . . 9
2.2 Background & Related Work . . . 10
2.2.1 Eclipse Attacks (EA) . . . 10
2.2.2 Localized Eclipse Attacks (LEA) . . . 11
2.2.3 Topology-Aware Localized Eclipse Attacks (taLEA) . . . 11
2.2.4 Contributions: Detection & Mitigation . . . 11
2.2.5 Related Work . . . 11
2.3 System Model . . . 12
2.3.1 Overlay Network Model . . . 12
2.3.2 P2P Protocol Model . . . 12
2.3.3 Lookup Mechanism . . . 13
2.4 LEA Attacker Models . . . 14
2.4.1 Fake Destination Attacker Behavior (FD-LEA) . . . 14
2.4.2 Pollution Selection Attacker Behavior (PS-LEA) . . . 14
2.4.3 Mixed Attacker Behavior (FD-PS) . . . 15
2.5 PASS: Divergent Lookups P2P Address Space Slicing . . . 15
2.5.1 PASS Preliminaries . . . 16
2.5.2 divPASS susceptibility to LEA . . . 16
2.6 Detection Mechanisms . . . 16
2.6.1 Lookup Result Voter Mechanism . . . 17
2.6.2 DMV Operation . . . 17
2.6.3 Lookup Reply Investigation . . . 17
2.7 Evaluation . . . 18

2.7.1 Simulation Environment 18
2.7.2 Simulation Workload Model - Fully Distributed Application 19
2.7.3 Simulation Churn Models 19
2.7.4 Simulation LEA Model 19
2.7.5 Evaluation Metrics 19
2.7.6 Case Study 1: LEA impact on divPASS using FD behavior 20
2.7.7 Case Study 2: Voting mechanism DMV against LEA attacks 21
2.7.8 Case Study 3: LEA impact using weighted FD-PS impact 22
2.7.9 Case Study 4: Detection and mitigation mechanisms response against weighted FD-PS behaviors 24
2.8 Conclusion 26

3 Structured: LA-Eviction 27
3.1 The Necessity of Eviction 27
3.2 Related Work 28
3.3 Localized Attack Model 29
3.3.1 Malicious Resources 30
3.3.2 Malicious Placement 30
3.3.3 Adversarial Behaviors 30
3.4 Eviction Mechanism 31
3.4.1 Detection Process 32
3.4.2 Removal Process 33
3.5 Evaluation 38
3.5.1 Simulation environment 38
3.5.2 Simulation model 39
3.5.3 Evaluation Metrics 39
3.5.4 Case Study 1: LA Impact 39
3.5.5 Case Study 2: EM Evaluation 41
3.5.6 Summary 43
3.6 Conclusion 43

4 Structured: RTP Attacks 45
4.1 Defending against RTP Attacks 45
4.1.1 Contributions: 46
4.2 Related Work: Typical Attacks & Mitigation Approaches 46
4.2.1 Contemporary Approaches 47
4.3 Routing Table Poisoning (RTP) 50
4.3.1 RTP Attacks Types and Targets 50
4.3.2 Attacker Capabilities 51
4.3.3 RTP Adversarial Behaviors 52
4.4 Detection Mechanism 53
4.4.1 Modified lookup Approach 54
4.4.2 Dynamic Majority Voter (DMV) 55
4.5 Sanitizing Mechanism (SM) 55
4.5.1 Forming a Quorum 56
4.5.2 Quorum Investigation 59
4.5.3 Reaching an Agreement 60
4.5.4 Malicious Removal Procedure 61
4.6 Evaluation 62
4.6.1 Simulation environment 62

4.6.2 Simulation Model . . . 62
4.6.3 Evaluation Metrics . . . 63
4.6.4 Experiment 1: RTP attack severity . . . 63
4.6.5 Experiment 2: SM Influence . . . 65
4.6.6 Summary . . . 68
4.7 Conclusion . . . 69

5 Super-unstructured: OEAs 71
5.1 Motivation: OEA and Super-P2P Overlays . . . 71
5.1.1 Contributions: . . . 72
5.2 System Model . . . 73
5.2.1 Overlay Model . . . 73
5.2.2 Attack Model . . . 73
5.3 Background & Related Work . . . 75
5.3.1 Attack Detection Mechanisms . . . 75
5.3.2 Anonymous Auditing of Node Degrees . . . 75
5.4 Detection Mechanisms . . . 76
5.4.1 Proactive Detection . . . 77
5.4.2 Reactive Detection . . . 79
5.5 Analysis . . . 80
5.5.1 Effectiveness of the Degree Bound . . . 81
5.5.2 Effectiveness of the Quorum . . . 83
5.6 Evaluation . . . 85
5.6.1 Simulation Framework . . . 85
5.6.2 Simulation Set-up and Metrics . . . 85
5.6.3 Case Study 1: OEA impact . . . 86
5.6.4 Case Study 2: Detection assessment . . . 88
5.7 Conclusion . . . 91

6 Streaming: Cheating Attacks 93
6.1 Motivation: Streaming & Cheating Attacks . . . 93
6.1.1 Contributions: . . . 94
6.2 Internal Attack Model . . . 95
6.2.1 Target, budget and placement . . . 95
6.2.2 *Drop-chunk* adversarial behavior . . . 96
6.3 Detection Mechanism . . . 96
6.3.1 Mechanism Overview . . . 96
6.3.2 Detection Trigger . . . 98
6.3.3 Processing a Detection Request . . . 98
6.3.4 Filing and Processing a Complaint . . . 98
6.3.5 General Notes . . . 100
6.4 Analysis . . . 100
6.4.1 Falsely Accusing Benign Headnodes . . . 100
6.4.2 Retaining Malicious Headnodes . . . 101
6.5 Evaluation . . . 102
6.5.1 Simulation Framework, Parameters and Metrics . . . 102
6.5.2 Case 1: *Drop-chunk* Severity . . . 103
6.5.3 Case 2: Detection Mechanism Performance . . . 104
6.6 Related Work . . . 106
6.7 Conclusion & Future Work . . . 107

7 Summary and Conclusions **109**
7.1 Attacks on P2P Systems 109
7.1.1 Structured Overlays 110
7.1.2 Unstructured Overlays 111
7.1.3 Streaming Overlays 111
7.2 Summary: Research Questions & Contributions 112

List of Figures

1.1 P2P overlays addressed in this thesis 2
1.2 Attacks investigated per overlay 5

2.1 Acceptance cases for the DMV. 18
2.2 FD-LEA impact on divPASS using different MP. 21
2.3 Combined divPASS/DMV performance with different values for MP. 23
2.4 Baseline results for FD-PS without detection. 24
2.5 Detection performance of FD-PS based LEAs. 25

3.1 Eviction process overview 31
3.2 Detection procedures 32
3.3 Removal Procedures 35
3.4 LA impact 41
3.5 EM performance 44

4.1 Detection procedures. 54
4.2 Technical aspects of SM procedures. 57
4.3 Example of types of peers in Q. 58
4.4 RTP attack impact. 64
4.5 SM performance measurements. 67
4.6 MRT decay due to SM 67
4.7 Ratio of types of peers in quorum Q. 69

5.1 Proactive and reactive detection mechanisms 78
5.2 OEA impact on super-unstructured networks 87
5.3 Detection mechanism performance 89
5.4 Comparing theoretical and simulation results $(k, l) = (6, 3)$ 90
5.5 Evaluating parameters for detection mechanism 91

6.1 Detection process for *Drop-chunk*. S denotes the source. 97
6.2 Attack's impact on DONet 104
6.3 Detection mechanism performance 105

7.1 Structured overlays: LA 110
7.2 Structured overlays: RTP 111
7.3 Unstructured overlays: OEA 111
7.4 Streaming overlays: BM data dissemination cheating attacks . . 112

List of Tables

2.1 LEA detection simulation parameters 19

3.1 LAs eviction: acronyms description 33
3.2 LAs eviction simulation parameters 38

4.1 RTP attacks & defenses: acronyms description 56
4.2 RTP attack & defenses: simulation parameters 63

6.1 Detecting cheating attacks: acronyms description 97

Chapter 1

Introduction

Peer-to-Peer (P2P) computing is an established paradigm used across a variety of data dissemination and data discovery applications. P2P networks gained a wide popularity due to utilizing the decentralized coordination to provide scalability, reliability and fault tolerance, which naturally leads to its wide applicability nowadays. In fact, P2P protocols increasingly constitute the foundations for many large-scale applications due to the inherited distributed nature of P2P systems. Originally applied for file sharing applications, it is increasingly utilized for diverse large-scale networked applications resulting in conquering a vast share of the current internet traffic with diverse data dissemination and data discovery applications such as file sharing, multimedia streaming, VoIP, online gaming, machine-to-machine communication, IoT and many others [GK03; RPI12; FTT10; AAM+15; GE12; WK13].

In order to support such fundamentally distinctive applications in terms of scalability, QoS and low overheads, P2P systems[1] correspondingly differ according to the application platform, i.e., a file sharing application requires a different overlay functionalities and data dissemination techniques than a video streaming application. However, the main common aspect between different P2P systems is that peers have only a partial view of the network as obtained from their neighboring peers, i.e., there is no large-scale P2P overlay where all peers are aware, or directly connected to, each other due to the CPU resources and storage space limitations. Other than the aforementioned characteristic, P2P overlay models differ remarkably in various aspects. Therefore, it is important to start by highlighting the fact that:

> *P2P overlays are different; they can not be addressed as a single entity. When security aspects of a P2P overlay is investigated, the specifics of the targeted overlay are critically important as the attackers always seek for a tiny vulnerability in the system to exploit.*

In that context, to present the flow of the work discussed in this thesis on a high-level, we start by emphasizing the main aspects that define the uniqueness of P2P overlays.

[1] Throughout the thesis, the terms system, network and overlay denote the same meaning and are used interchangeably

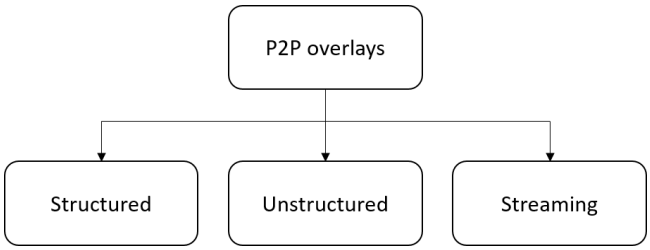


Figure 1.1: P2P overlays addressed in this thesis

1.1 P2P Overlays

The aim of this section is to discuss the main differences between various P2P overlays which highlight the need to address security threats differently depending on the targeted overlay model. As depicted in Figure 1.1, we target the three major existing P2P overlays: structured, unstructured and streaming overlays, as these overlays constitute the basic concepts behind the most widely distributed applications [**emule**; **steinheimer2013p2p**; **6047218**; RPI12; AAM+15]. To that end, we now highlight the main factors that differ in each P2P overlay.

Differences on a high-level

Each P2P overlay model constitutes an independent system model that defines how peers:

1. are assigned IDS.
2. contact each other.
3. disseminate data across the designated overlay.
4. share responsibilities and priorities according to the overlay protocol, i.e,. peers might acquire different responsibilities and criticality due to (i) possessing a trending content, (ii) being positioned near the data source, (iii) being responsible to distribute data to a certain fraction of peers (for example, in super-unstructured systems), or (iv) being more trusted to other peers, i.e., in a trust-based system, which is a common approach in various P2P overlays.

In the following, we briefly discuss the aforementioned aspects with respect to the different overlay topologies considered in the thesis.

Structured overlays:

Structured overlays are distinguished by the fact that peers can infer the closeness of other peers by their IDs. Peers are assigned IDs such that the distance between peers are calculated through the fraction of shared prefix in their unique identifier, i.e., peers can calculate how close they are to each other. Hence, peers usually make use of convergent forwarding approach of messages to contact each other which provides a very efficient searching mechanism for either peers or content. However, despite the attractiveness and efficiency of such a feature,

the susceptibility of structured overlays to attacks increases as the attacker is capable of (a) revealing the location of other peers, (b) revealing the distance to other peers, and (c) tracing the forwarding path of a specific message.

Unstructured overlays:

Unlike structured overlays, peers in unstructured overlays are randomly positioned in the overlay, i.e., no protocol that arranges how peers are close or far from each other. This denotes that in an unstructured overlay, peers either flood their neighbors with requests searching for a specific peer, or a super-unstructured topology is applied, i.e., some peers are promoted to *super peers* and thus, become responsible for forwarding messages to a certain group attached to this peer to allow for more organization of the overlay and less flooding of requests. Accordingly, compared to structured overlays, unstructured overlays are easier to maintain, however, on a large-scale overlay, looking up a peer becomes very costly in terms of time, resources and overhead.

For this reason, most large-scale unstructured overlays are coupled with a super-P2P topology to enhance the system's performance in terms of connecting peers and forwarding messages in a more efficient manner, which is referred to as super-unstructured overlays. Thus, the set of super peers in the system acquire higher priorities and responsibilities in the overlay. In turn, grab the attackers' attention given their critical role in the overlay.

Streaming overlays:

Streaming overlays are mainly characterized by the existence of a source peer, i.e., the peer that owns the stream and is responsible for disseminating the stream chunks to the rest of the overlay. Other peers who join the stream are assigned random IDs and contact each other mainly in either tree or mesh-based fashion.

Peers disseminate data via advertising for the existing data in their buffer and thus, other peers can request certain data elements. Notably, as a fraction of peers starts to receive the stream chunks from the source, they in turn participate in the data dissemination process via relaying the received content which highly reduces load on the source peer and increases the system's reliability. Specifically, peers disseminate data via advertising for the existing data in their buffer and thus, other peers can request certain data elements. Although the efficiency and reliability provided by the existing data dissemination techniques, various attacks focus on abusing those techniques mainly through aiming at *cheating* other peers when advertising about the availability of the stream chunks in their buffer or through the actual transmission of the stream chunks.

Interpretation

As discussed above, we emphasize how P2P overlays do not operate in the same fashion. In fact, despite sharing the features of distributiveness and scalability, each P2P overlay notably differs in how peers behave within the overlay and how the overlay itself is organized. To that end, investigating the core question of this thesis "How secured are P2P systems" can be only investigated when

dealing with each P2P overlay's aspect separately. Therefore, it is important to reflect the following statement:

> *Indeed, the aforementioned design practices render P2P networks susceptible to various attacks. An attacker conducting an attack on a certain P2P overlay targets exploiting one of those characteristics.*

Note that the aforementioned aspects regarding each system model are described in details in the corresponding chapter(s).

1.2 Attacks on P2P Overlays

Over the last years, various attacks on P2P overlays evolved and still mutate, which we refer to throughout the thesis as progressive attacks. For example, eclipse, poisoning, cheating, flooding, sybil and publishing are examples of the most known attacks that (a) exist, (b) are easily conducted, and (c) are continuously evolving on P2P overlays [SCD+04; STR10b; ZJT13; KLR09; LMS+10; Dou02; GRS+14; SND+06; LYL14; IRS18b; IRS18a; KLK+12]. Those attacks are overlay-specific, i.e., they are conducted through exploiting a single characteristic in a given P2P overlay. Hence, mitigating such attacks becomes more challenging as no single or generic defense scheme provides a thorough protection to various P2P overlays due to the different functions and aspects deployed by peers in each overlay.

While designing a mitigation/detection/sanitizing scheme is desired to enhance the overlay's resiliency to attacks and effectively increase the users' trust towards the provided service, various constraints arise that can exorbitantly entail service degradation as:

(1) schemes that constitutes a central monitoring/ decision-responsible entities conceal the attractive features and data dissemination efficiency of distributed P2P overlays. Moreover, such central schemes the overlay's resiliency due to the additional single point of failure and the elevated probability that attackers target taking over such entities.

(2) complex cryptographic schemes, which are oftentimes proposed in the literature for P2P systems, are unfeasible to be processed by light-weight computing devices. Thus, decrease the overall service quality provision.

(3) schemes that are tailored specifically against a single attack variant, even from the same attack class, become useless when the attacker adapt to the mitigation scheme, e.g., by modifying a single malicious behavior conducted by malicious peers.

To this end, in this thesis, we address the susceptibility of various P2P overlays to different progressive attacks while carefully addressing the aforementioned potential drawbacks when designing a countermeasure.

> *Our focus in this thesis is on designing countermeasures* for different P2P overlays *that: (1) are attack-class generic, (2) are completely distributed, (3) do not depend on complex cryptographic schemes.*

Moreover, we propose more aggressive variants of the investigated attacks and accordingly analyze their impact as a prerequisite to design an effective mitigation/detection attack-class generic countermeasure. In a nutshell, this

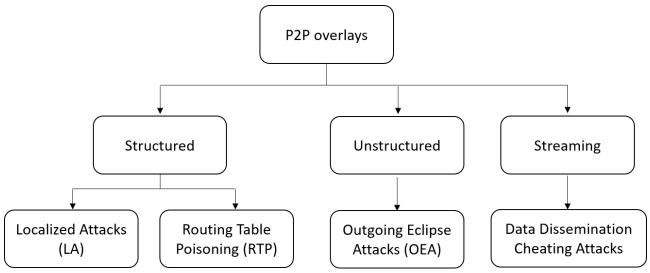


Figure 1.2: Attacks investigated per overlay

thesis focuses on analyzing the impact of current progressive attacks on different P2P overlays and developing a countermeasure that can effectively detect, and thus mitigate, the attack's conducted adversarial behavior.

Specifically, in the context of the aforementioned challenges, as depicted in Figure 1.2, the following attacks are investigated on the corresponding overlays:

Localized Attacks (LA)

LAs are conducted on ***structured*** overlays and tackle the closeness-awareness key factor of such overlays. In other words, the attacker abuses this factor and targets specific peers to conduct his malicious behavior. We show that the generic (LA)s, with more focus on the class of Localized Eclipse Attacks (LEA)s, are feasible and indeed severe on structured overlays.

Routing Table Poisoning (RTP)

Similarly, such attacks target ***structured*** overlays, where malicious peers aim at conquering benign peers contact lists. We prove in our work, specifically in Chapter 4, that RTP attacks are a higher-level form of LA attacks. This denotes that by successfully conducting an RTP attack, LAs, Sybil, indexing or other forms of attacks that can be conducted on structured overlays become feasible.

Outgoing Eclipse Attacks (OEA)

In this part of the thesis, we propose OEAs, where malicious peers target eclipsing *outgoing* messages from benign peers. Unlike structured overlays, where specific peers can be conveniently attacked due to the closeness notion, OEAs are conducted using both scenarios, i.e., (a) where every benign peer is attacked, and (b) the attacker targets only a specific set of peers. Regarding (b), such attack behavior is of interest as we consider the super-unstructured variant of ***unstructured*** overlays, as described in Section 1.1. In a nutshell, we explore and validate the impact and feasibility, respectively, of conducting OEAs on super-unstructured overlays.

Data Dissemination Cheating Attacks

In ***streaming*** overlays, as mentioned in Section 1.1, data dissemination techniques are a key factor given the hard delay and overhead constraints of sending and receiving streaming data. Attacks that target exploiting, and thus perturb,

such functionalities are continuously evolving. In this part of the thesis, we investigate the severity of a composite data dissemination attack and propose a counter measure, where malicious peers are able to be placed in critical positions in the streaming overlay before conducting several adversarial behaviors to disrupt the data dissemination technique used.

To that end, we detail the research questions (RQ) we address through out the thesis and the corresponding contributions.

1.3 Research Questions and Contributions

To that end, our focus in this thesis is on answering the following research questions, which consequently constitute the contributions as follows.

Research Question (RQ1): How resilient are structured P2P overlays to attacks

In this research question we aim at investigating the impact of different attacks on structured P2P overlays in order to assess the necessity of designing a detection and eviction countermeasures. We specifically address two attacks:

Eclipse Attacks (EA): In Chapter 2 we address Eclipse Attacks (EA)s and especially the class of Localized EA's (LEAs), i.e., the attacker directs all of his resources to eclipse a specific set of peers. Subsequently, our contribution is three-fold as follows.

Contribution (C1): Securing structured overlays against LEAs

- Assessing the severity of variant LEA attacks when existing mitigation techniques are functioning.
- ***(C1.1)*** Developing a highly accurate mechanism to detect malicious peers when they target a specific peer to attack. This work was published at ICPADS 2015 [IGS15].
- ***(C1.2)***Developing an eviction mechanism that is generally applicable to various LA types while effectively evicting malicious peers from the overlay. This work was presented at CNS 2016 [IGS16].

Routing Table Poisoning (RTP) attacks: An RTP attack is defined as the act of malicious peers aiming to exist in benign peers routing tables to conduct an attack in a structured overlay, i.e., poison benign peers Routing Table (RT). In this work, we address the severity of poisoning benign peers RT and accordingly propose a defending mechanism. This work was presented at the Journal of Computers & Security 2017 [IGS17]. We detail our contribution in the following.

Contribution (C2): Sanitizing benign peers RT from malicious peers conducting RTP attack

- Evaluating the severity of RTP attacks.

- Developing an adaptable RTP attack mitigation approach. Thereby, we propose a protocol-independent, fully distributed, simple and effective detection and overlay-sanitizing mechanism.

Research Question (RQ2): How to defend against Outgoing Eclipse Attacks (OEAs) conducted on super-unstructured P2P systems.

We show that the class of Outgoing Eclipse Attacks (OEAs) are particularly threatening for super-unstructured systems, specifically given the criticality and the small fraction of super peers in such systems. Accordingly, we highlight the necessity for developing an effective detection and expulsion scheme to restore the system's reliability and availability. We published this work at TrustCom 2018 [IRS18a]. To that end, our contribution towards defending against OEAs is:

Contribution (C3): An effective detection and peer eviction mechanism to mitigate OEAs in super-P2P systems.

- Assessing the impact of OEAs on super peers in unstructured overlays.
- Developing an adaptable RTP attack mitigation approach. In other words, we propose a composite proactive and reactive mechanism that mitigates the effect of routing table infiltration as well as the subsequent OEAs.

Research Question (RQ3): Online data streaming overlays: what is the impact of internal data dissemination cheating attacks on the streaming quality and how to mitigate such an attack in a tightly QoS constrained environment

Our focus here is to address a specific data dissemination cheating attack variant, where (a) malicious peers are able to occupy the most vital positions in the overlay to maximize the perturbations, and (b) detecting such an attack is very challenging for benign peers. In order to do so, we address the severity of data dissemination cheating attacks on the stream quality, i.e., peers satisfaction from the streaming service.

Contribution (C4): Designing a detection mechanism to counter the impact of internal DoS attacks.

This work was presented at WoWMoM 2018 [IRS18b]. Our contribution to this research question constitutes of the following.

- Investigating the effectiveness of internal cheating attacks.
- Proposing a detection mechanism that ensures peers satisfaction regarding the streaming quality.

1.4 Thesis Structure

For each overlay type, the system model, the corresponding attack model and the proposed countermeasure is presented in the designated chapter. In Chap-

ter 2 and Chapter 3, we describe the detection and eviction mechanisms, respectively, designed for LEA attacks on structured overlays. Chapter 4 presents the evaluation of RTP attacks on structured overlays along with describing the developed sanitizing mechanism against RTP attacks.

Afterwards, in Chapter 5 we describe our two-fold proactive and reactive mechanisms to defend against routing table infiltration on super-unstructured overlays. Chapter 6 addresses the impact of cheating attacks on online P2P streaming overlays and proposes a detection scheme to preserve the streaming service quality. Finally, Chapter 7 contains the thesis summary and the subsequent conclusions.

1.5 Publications

The following publications, in parts verbatim, are used in this thesis:

- H. Ismail, S. Roos, and N. Suri. "A Composite Malicious Peer Eviction Mechanism for Super-P2P Systems". In: *IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom)*. 2018
- H. Ismail, S. Roos, and N. Suri. "A Detection Mechanism for Internal Attacks on Pull-based P2P Streaming Systems". In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2018
- H. Ismail, D. Germanus, and N. Suri. "P2P Routing Table Poisoning: A Quorum-based Sanitizing Approach". In: *Computers & Security* 65 (2017), pp. 283–299
- H. Ismail, D. Germanus, and N. Suri. "Malicious Peers Eviction for P2P Overlays". In: *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*. 2016, pp. 216–224
- H. Ismail, D. Germanus, and N. Suri. "Detecting and Mitigating P2P Eclipse Attacks". In: *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*. 2015, pp. 224–231

The following publications are affiliated to various aspects discussed in this thesis, nonetheless, they are not included:

- D. Germanus, H. Ismail, and N. Suri. "PASS: An Address Space Slicing Framework for P2P Eclipse Attack Mitigation". In: *IEEE Symposium on Reliable Distributed Systems (SRDS)*. 2015, pp. 74–83
- K. Demir, H. Ismail, and T. Vateva-Gurova. "Securing the Cloud Assisted Smart Grid". In: *Internal Journal of Critical Infrastructure Protection* (2018)

Chapter 2

Structured Overlays: Detecting LEAs

The work presented in this chapter, published in [IGS15], counts towards addressing Research Question 1 and the corresponding sub-contribution ***(C1.1)***. As mentioned in Contribution 1, this work is two-fold. In this chapter, we present our detection mechanism for detecting LEA attacks. Secondly, in Chapter 3 we build upon this detection mechanism to design an eviction mechanism to efficiently expel malicious peers from the system using a sophisticated form of LEA attacks. Hence, although the system model holds, we present our sub-contributions differently due to the different progressive attacks we counter.

We start by briefly highlighting the problem which motivates the necessity of tackling the security threat induced through LEAs on structured overlays. Afterwards, in Section 2.2 the technical background and related work are presented. Section 2.3 describes the system model and the concepts underlying the technical sections covering the attacker model (Section 2.4), divergent lookups (PASS) (Section 2.5), and detection mechanism (Section 2.6). Finally, the attack severity, mitigation efficiency, and detection rates are evaluated in Section 2.7.

2.1 Problem Statement

P2P networks inherently provide good fault-tolerance due to their design approach of redundant message exchange and replicated data storage. Moreover, the decentralized protocol design requires only partial views of the network and thereby facilitates their scalability. Yet, the partial view of each peer on the P2P overlay network also introduces susceptibilities to various attacks [LMG+10; WTC+08; Dou02; SCD+04; CCF+12].

This work focuses on Eclipse Attacks (EA) and especially the class of Localized EA's (LEA) that are known to have a significant impact on P2P functionality [SND+06; DKS12] of availability, integrity, and confidentiality. Moreover, no generic LEA mitigation technique has been found that also preserves the properties of scalability, decentralization, openness, and timeliness.

Recently, divergent lookups have been proposed as an effective mitigation technique for a variant of localized EA, known as **Topology Aware LEA**

(taLEA), [GRS+14; GIS15], where the attack efficiency stems from very selective placement of malicious peers in the victim peers' vicinity. taLEA depends on knowledge of the topology and the overlay protocol to place very few but very carefully placed malicious peers to cause damage. Unlike the niche taLEAs, LEAs form the more general attack case with malicious peers scattered all over the overlay network. To that end, we can describe how LEA attacks are more: (i) feasible to conduct and (ii) severe than taLEA in the following statement.

> *While more malicious peers are required to achieve an impact comparable to a focused taLEA scenario, the generalized topology agnostic placement of peer nodes in a LEA makes for a very easy-to-conduct high-damage attack, and hence the need for LEA mitigation.*

Contributions towards Detecting LEAs

On this background, the contributions in this work are (i) demonstrating limitations of conventional divergent lookup mechanisms resilience to LEAs under sophisticated attacker behavior scenarios, and (ii) enhancing divergent lookups via the development of a highly accurate mechanism to detect malicious peers which is based on a dynamic voting algorithm. This enables divergent algorithms not only to mitigate selected LEA variants, but also the generic LEA attack cases. We have conducted comprehensive simulation experiments to assess our contributions considering a diverse P2P parameter landscape. Our approach shows divergent lookup mitigation effectiveness of up to 96% in LEA scenarios involving up to 25% malicious peers in the overlay network. Moreover, we are able to detect close to 100% of malicious peers where the detectability varies depending on the exact attacker behavior.

2.2 Background & Related Work

A variety of EA mitigation techniques have been proposed, yet their effectiveness as a countermeasure for localized attacks is either quite limited or violates P2P aspects of scalability or decentralization. We first provide a high-level overview on relevant EA variants, followed by an overview of our mitigation approach, and a discussion on related work.

2.2.1 Eclipse Attacks (EA)

The goal of an EA [SCD+04] is to *eclipse* resources (peers or data managed by peers), i.e., prevent benign peers' service provision or provide nefarious services by using a set of malicious peers. Peers targeted to be eclipsed are referred to as victims $v \in V$. A variety of approaches can be taken to launch EAs, and in many cases the decentralized routing mechanism is attacked. Malicious peers may collude and behave inconsistently which further complicates their detection. Localized EA (LEA) and topology aware EA (taLEA) are two common variants of EAs which are discussed next.

2.2.2 Localized Eclipse Attacks (LEA)

LEAs [SND+06] are a subcategory of EAs that eclipse only a subset of peers in the P2P overlay network where malicious peers are scattered through the overlay. The adversary chooses the subset, for example, based on the resources managed by the peers to be attacked.

2.2.3 Topology-Aware Localized Eclipse Attacks (taLEA)

Topology-aware LEAs [DKS12; GRS+14] (taLEA) are a specialized LEA variant which require only a small, fixed amount of malicious peers to launch an efficient attacks against overlay networks of arbitrary sizes. To this end, the adversary places malicious peers at specific locations in the overlay network's topology. In our previous work in [GRS+14; GIS15], we managed to mitigate taLEA using divergent mechanisms which are described in Section 2.5.

2.2.4 Contributions: Detection & Mitigation

In this work, we demonstrate three sophisticated LEA variants in Section 2.4. To mitigate these attacks, we assess divergent lookups [GRS+14] for their suitability in the new attack's context. We highlight how divergent algorithms are unsuitable to mitigate LEAs and how severely does the proposed adversarial LEA behaviors negatively impact the overlay stability in terms of reliability and connectivity between peers. Section 2.5 presents the main concepts of (a) divergent lookups, and (b) the P2P Address Space Slicing Technique (PASS) [GIS15], highlighting the threats that exist from a LEA. In Section 2.6, we introduce the technical foundations for mitigating generic LEAs and detecting malicious peers that conduct the aforementioned LEA variants. Finally, we evaluate our approach in regard to the lookup reliability and performance in Section 2.7.

2.2.5 Related Work

In [Dou02], Sybil attacks were introduced where the attacker can launch an attack with a small set of malicious peers and can consequently garner multiple addresses which allows malicious peers to fake being a larger set of peers. Using Sybil attacks, authors in [KLR09] launched a LEA via a chain of Sybil/malicious nodes. However, the attack relies on the strong assumption about the existence of a single path towards the victim. In [SEB07], a LEA is launched using Sybil peers. Although the authors proposed a mitigation scheme, the scheme is based on a *centralized* encryption authority. Using the same concept, authors in [CDG+02] proposed adding Certificate Authorities to peers' network IDs while joining the network. Although authors in [FMR+09] proposed a mitigation scheme based on preventing malicious entities from selecting their own network IDs, the mitigation scheme is based on a signing entity that uses public key cryptography.

In [BM07], a mitigation mechanism is proposed based on assigning multiple paths for each lookup using disjoint paths. Nonetheless, a cryptographic scheme is used that can only be substituted by a centralized authority. In addition, in [OC01] a similar approach is proposed. However, messages overhead due to using multiple paths is not addressed.

Similarly, the authors in [LMS+10] highlight how publish attacks could be used to attack the KAD network which is a Kademlia based network through flooding peers' index tables close to the victim with false information which is a simplistic taLEA variant. However, they didn't provide a mitigation scheme.

In [CCF+12], a KAD network crawler is introduced to monitor the network status and detect malicious peers during a LEA. However, in a distributed P2P system, a high overhead arises if each peer uses such a mechanism to detect malicious entities. This becomes impractical as the overlay size increases.

2.3 System Model

This section presents the system model for structured overlays that is used for the evaluation of our approaches in Contributions 1 and 2. Utilizing the established models from [IGS15; IGS16; IGS17], we start by describing the overlay model topology before presenting the main protocol model aspects.

2.3.1 Overlay Network Model

The network is modeled as a directed graph $D = (P, E)$. P is the set of peers $p \in P$ in the overlay network. Distinct peers $p, q \in P$ that maintain a neighbor relationship are represented by $e = (p, q) \in E$.

We further partition P as follows: benign peers B, malicious peers M and victim peers V, so that $P = B \cup M$, where $B \cap M = \emptyset$, $V \subseteq B$ and $N = |P|$, where N is the overlay size. Malicious peers $m \in M$ refer to peers being controlled by an attacker and may behave maliciously. Peers targeted by the attacker are victims $v \in V$. Furthermore, malicious and victim peers do not churn, which in fact gives the attacker more control over the available resources.

Peers $b \in B$ show benign behavior in the network, i.e., according to the P2P model specification and no adverse intentions.

Poisoned peers $o \in O$ refer to benign peers that store or propagate malicious information as a consequence of contacting malicious peers, where $O \subseteq B$. Churning peers $c \in C$ refer to peers that leave the network either randomly or according to a certain distribution.

As only benign peers, except victim peers, experience churning behavior, $C \subseteq B$ and $V \cap (O \cup C) = \emptyset$.

2.3.2 P2P Protocol Model

Our abstraction for structured P2P protocols consists of five salient aspects as detailed below.

Address Space

Peers have a unique assigned identifier referred to as the peers' *keys*. Typically, keys are generated from an external feature such as the IP address, MAC address, a serial number, or a random number. Keys usually have a length of $w \in \{128, 160, 192\}$ bits and are mapped onto the overlay's *address space* which is used to address resources such as peers and addressable data tuples.

Distance Function

A distance function is defined for peers on the address space. The distance notion is an important feature for many peer operations and the choice of the distance function differs among P2P protocol implementations. For example, Kademlia [MM02] uses the XOR operation to calculate the common prefix length (CPL) using the bit-string representation of the keys from two peers.

Routing Table (RT)

Each peer maintains an RT that contains contact information about neighboring peers. Contact information is a tuple that relates keys of peers with their underlay network information (e.g., IP address and port number). Routing tables vary among protocols and usually store k contact information tuples of peers in w lists for distance ranges $[2^i, 2^{i+1})$ with $i = 0 \dots w-1$, and k constant. In order to resolve new contact information a lookup call is initiated.

Proximity

Each peer defines a proximity area, typically a proximate and sparsely populated region of the address space that is selected based on the overlay size N and the key length w. We define the proximity of a peer as the set of peers with the closest distance to this peer, and subsequently stored in its RT.

2.3.3 Lookup Mechanism

In case the destination peer p_v for a specific message to be sent by peer p_i is not stored in p_i's routing table, a lookup call is initiated to *resolve* p_v's contact information. To initiate a lookup, p_i selects α peers from its RT to query them about p_v. We now describe the two main lookup mechanisms used in structured P2P overlays.

Convergent Lookups

A commonly applied design best practice are *convergent lookups*, i.e., peer p_i selects a set of known peers with closest possible distance to p_v, and iteratively queries each of them to either return the contact information or to repeatedly *forward* p_i's lookup request to even closer peers until p_v can either be resolved or the lookup is dropped due to a timeout. Due to the structured nature of the overlay, the convergent mechanism guarantees low message overhead with minimum number of hops for resolving a certain lookup. Nevertheless, selective placement of malicious peers in a very close distance to the victim eclipses the victim's existence as evaluated in [DKS12].

Divergent Lookups

We proposed in [GIS15], *divergent lookups* to mitigate attacks that make use of convergent mechanisms. Divergent lookups restrict the ability to contact peers close to the victim, where the notion of closeness is referred to as the peer's proximity. In [GIS15], the PASS algorithm efficiently defines the address space range that contains peers with high probability of resolving the contact

information of p_v. However, contacting peers during lookups from different address space ranges naturally results in suboptimal performance and reliability degradation. Unlike a convergent mechanism, which is highly susceptible to certain localized attacks, divergent mechanisms show high resiliency to such attacks while providing a comparable performance to convergent schemes. As we build upon PASS in the detection mechanism presented in Chapter 2, PASS is described in details in Section 2.5.

2.4 LEA Attacker Models

We propose a new attack model based on behavioral patterns of malicious peers. The attack model builds upon and extends LEA (cf. Section 2.2.2), thus, all the attack behaviors discussed here represent the generic LEA.

To launch the attack, malicious peers $m \in M$ join the overlay and we assume they are uniformly distributed across the address space. Once a peer m receives a lookup request for the victim peer, different *attacker behaviors* can be activated. Moreover, the proposed LEA based behaviors are chosen based on security goals (availability, integrity, confidentiality) that exploit the lookup mechanism. Next, we introduce three new complex attacker behaviors that collectively represent the generic LEA behaviors:

2.4.1 Fake Destination Attacker Behavior (FD-LEA)

In FD-LEA, malicious peers fake the victim's identity, which threaten the availability, confidentiality and exploit the inherent partial view of each peer.

Technical Description

During a lookup, once a malicious peer receives a lookup request for a victim peer, it replies to the lookup initiator p_r with contact information that points to a malicious peer that fakes owning the key p_r is looking for.

Behavior Discussion

The overlay's reliability is severely affected since the lookup call terminates once a malicious peer returns a fake destination and p_r believes that the reply was sent from a benign peer that holds v's contact information. Consequently, the availability of the victim peer's service provision is negatively affected. Moreover, in case of unencrypted message payloads, the confidentiality would also be affected, as p_r sends its message to the colluding malicious peer that may subsequently inspect it.

2.4.2 Pollution Selection Attacker Behavior (PS-LEA)

In a PS-LEA behavior, malicious peers reply only with malicious contact information which threaten the availability and exploit the candidate selection mechanisms for the lookup initiator peer p_r. The main aim of the attacker during a PS-LEA is to pollute p_r's candidates selection queue which is maintained over the different lookup iterations to store contact information of peers that

may be queried. Lookup iterations refers to the number of rounds where p_r sends parallel lookup requests to different peers requesting v's contact information.

Technical Description

Initially, p_r stores a list that contains all the possible candidates that could be queried in the next iterations. This list is updated after each iteration from other queried peers that have no knowledge about v. The selected candidates set sent to p_r are selected according to the used lookup algorithm.

Behavior Discussion

Once peer m receives a lookup request for a victim peer, only colluding malicious peers located all over the address space are returned. Hence p_r contacts malicious peers in the next iterations until the lookup request times out after i_{max} iterations. Similarly, the availability of the victim peer's service provision is negatively affected.

2.4.3 Mixed Attacker Behavior (FD-PS)

The third attacker behavior is a combination of the previous two, where a probability parameter is the basis for a switching decision between the proposed adversarial behaviors. Such a sophisticated attacker behavior has not been considered in previous work [DKS12; GRS+14] so far.

Technical Description

For mixed FD-PS LEA behavior, the attacker chooses weights for the probability of either behavior to be active, and behaviors may be subject to a switch in-between different lookup iterations.

Behavior Discussion

The impact of this behavior on the victim peers is the same as discussed before for the individual attack behaviors. However, activating both attacks with different weights helps vary the degree of perturbation that can be caused by each individual attack.

2.5 PASS: Divergent Lookups P2P Address Space Slicing

Divergent lookups have been proposed as a suitable taLEA mitigation technique in [GIS15; GRS+14]. In a nutshell, divergent lookups avoid searching the destination peer's proximity to skip out on querying malicious peers under taLEA assumptions. Also, divergent lookups match the mitigation requirements described beforehand. In this work, we assess the mitigation potential of divergent lookups for the more generic LEA variant. We briefly describe divergent lookups [GRS+14; GIS15].

2.5.1 PASS Preliminaries

Divergent lookups segregate the address space into *CPL slices*, i.e., creating equivalence classes according to the CPL peers share with the destination. The technique is called *P2P address space slicing* (PASS) and requires two more threshold parameters, namely upper t_u and lower threshold t_l. We define $0 \leq t_l \leq t_u \leq t_p \leq w$ with t_p being the proximity threshold. Divergent lookups that make use of PASS, detailed in our previous work in [GIS15], try to resolve the destination's contact information from peers in the CPL slice interval $[t_l, t_u]$ because other intervals, as discussed at next, are suboptimal:

- $[0, t_l)$: This range contains a large amount of peers, divergent lookups in that range tend to yield a bad performance or even timeout.
- (t_u, t_p): This range contains so called *dead ends* which represent peers that cannot reach the destination, i.e., no path towards the destination based on contact information of neighbor peers can be found. Running divergent lookups in that range yields a low reliability.
- $[t_p, w)$: This range is populated with malicious peers, therefore to be avoided by the lookup. Otherwise, reliability would significantly decrease.

2.5.2 divPASS susceptibility to LEA

Nevertheless, in that context, launching FD-LEA, PS-LEA or mixed FD-PS LEA on the selected CPL range (t_u, t_p), can severely degrade divPASS performance and reliability. PS-LEA behavior can simply (i) send the set of malicious peers within the CPL as possible candidates to p_r, (ii) divert the set of possible candidates outside of the suitable CPL range selected by divPASS, or even (iii) divert the request towards *dead end* peers. Similarly, malicious peers launching FD-LEA block the request from reaching to benign peers within the selected CPL that might have an LDE towards the destination.

Although lookups may be executed in parallel to improve on fault-tolerance and timeliness, divergent lookups are still susceptible to LEA with its variants as evaluated in Section 2.7. Obviously, the set of results can differ due to several malicious and benign causes. Two detection mechanisms presented in the next section have been designed to deal with such inconsistencies and allow to identify malicious peers that conduct LEAs (with FD and PS attacker behaviors).

2.6 Detection Mechanisms

We propose two differing detection mechanisms: (i) a lookup result voter, and (ii) a lookup reply investigator. The first mechanism analyzes the result set after the lookup completion and detects LEAs with the FD behavior. The second detection mechanism assesses, after each iteration, the lookup's candidate list to detect LEAs with PS attacker behavior. Both variants have been integrated into divPASS and will be evaluated in the subsequent evaluation section.

2.6.1 Lookup Result Voter Mechanism

The obvious reasons for lookup result inconsistencies are overlay perturbations such as ongoing attacks, outdated routing table entries, or perturbations in the underlay network. In order to detect inconsistencies, we use a *dynamic majority voter* (DMV) [GSC91]. DMVs are used to assess a set of inputs and thus, decide whether a valid output exists or not, where valid denotes a non empty majority of matching inputs. Basically, using DMV allows to (i) ensure reliable lookup operation in perturbed overlays, and (ii) identify maliciously behaving peers. The DMV can process up to α different inputs, which we group in three classes: *correct*, *fake*, or *no LDE towards the victim*.

In the following subsection, we discuss the DMV's operation with a focus on the reliable selection process from an inconsistent result set.

2.6.2 DMV Operation

Initially, p_r initiates a divergent lookup with α parallel requests. Once a peer replies with an LDE towards the victim p_d to p_r, the lookup terminates and the result is evaluated. Nevertheless, to allow the DMV to process a set of results, we modified the divPASS algorithm such that lookups wait for maximum i_{max} iterations until up to α replies are available.

Due to the probability that no different α benign peers have LDEs towards p_v, a maximum of c correct replies are returned to p_r. Furthermore, a fraction f_m of malicious peers within the specified CPL might intercept the lookup, which in turn will return f_m fake replies to p_r. In addition, n_e empty replies can be returned back to p_r due to (i) dropped replies and (ii) the lookup request can neither be intercepted by a malicious peer nor a benign peer have an LDE to p_d until the maximum number of iterations i_{max} is reached. To that end, the maximum number of replies that can be passed as an input to the DMV is $\alpha = c + f_m + n_e$. The next step is that the α received replies are passed as inputs to the DMV, which in turn decides whether to accept or reject the results, as shown in Figure 2.1, based on the following cases:

1. $\alpha \geq 3$: the DMV checks if a majority of a valid results is available, i.e., either $c > f_m$ or $c < f_m$.
2. $\alpha = 2$: the voter returns the contact information of index α as a valid reply in case: (i) both replies are identical and (ii) reply $\alpha \notin R_e$, where R_e is the set of empty replies. Otherwise, the DMV rejects the results.
3. If $\alpha = 1$ and reply $\alpha \notin R_e$: the DMV returns the only available reply and assumes it to be valid.

2.6.3 Lookup Reply Investigation

As mentioned in Section 2.4, the lookup replies of malicious peers that conduct a LEA with PS attack behavior contain contact information about other colluding malicious peers. Accordingly, in order to mitigate such attack while providing a detection feature to such malicious adversarial behavior, we propose an additional detection mechanism to assess further lookup replies before inserting them into the candidate selection list for subsequent iterations.

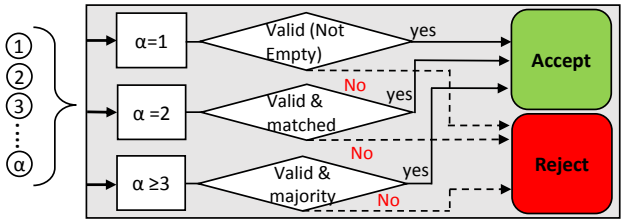


Figure 2.1: Acceptance cases for the DMV.

Technically, investigating received lookup replies, before inserting possible candidates in p_r's selection list, is based on (i) detecting peers whose replies point to peers outside of the CPL range $[t_l, t_u]$, and (ii) assuring that no candidates outside of the specified CPL can be inserted in the possible candidates list. Moreover, each peer is allowed only once in the candidate selection list. As a consequence, malicious peers that keep on sending malicious replies within the specified CPL range cannot excessively load the candidate selection list. At next, we assess our mitigation and detection schemes for the proposed attacks in a comprehensive simulation case studies.

2.7 Evaluation

In this section, we assess the performance and reliability of divPASS. To do so, we integrated it with our detection mechanism. For evaluation, we present four study cases:

1. **FD-LEA impact on divPASS:** evaluation of the impact of LEA using FD behavior on divPASS.
2. **Voting mechanism against LEAs:** divPASS resilience assessment after integrating the DMV as a mean to mitigate LEA launched with FD adversarial behavior.
3. **LEA impact on divPASS using FD-PS behavior:** evaluation of LEA impact launched with FD-PS attacker behavior on divPASS.
4. **Detection and mitigation mechanisms for FD-PS:** assessment on how our detection and mitigation techniques perform in a FD-PS behavior scenario during LEA.

Firstly, we detail the simulation environment, parameters, different models and metrics used throughout the experiments. Secondly, we present each of the above mentioned case studies, results and an interpretation of our observations.

2.7.1 Simulation Environment

Simulations were carried out using the OMNeT++ simulator [Pon93] and OverSim [BHK07] that provides various P2P protocol implementations for the simulator environment. In order to validate our results, each simulation was scheduled for 4 hours runtime and 12 repetitions were conducted to allow for confidence interval computation. The simulation parameters used in conducting experiments are presented in Table 2.1.

Parameter	Value	Parameter	Value
i_{max}	10	α	5
w	128	t_p	80
t_l	4	t_u	6
MP	$5\%, 15\%, 25\%$	Q_{max}	50

Table 2.1: LEA detection simulation parameters

2.7.2 Simulation Workload Model - Fully Distributed Application

In our simulation workload model, peers send lookup messages looking for random peers, on average every 10 seconds with a standard deviation of 5 seconds.

2.7.3 Simulation Churn Models

In our experiments, different churn models are used which are described below. Churn refers to the rate peers join and leave the overlay. **NoChurn:** refers to a static overlay where peers never leave the overlay once they have joined. **Pareto (P-7200):** Using the Pareto churn model, peers acquire an average lifetime and a dead time of 7200 seconds according to a Pareto distribution which gives a more realistic overview to real life scenarios [ZL06].

2.7.4 Simulation LEA Model

A central LEA parameter that we will refer to in the experiments' result discussion is **Malicious Peers per CPL (MP)**. It reflects the average number of malicious peers for a given divPASS CPL region. This metric provides insights about the severity of LEA attacks for an increasing amount of malicious resources. Data collection occurs at periodic intervals for each simulation run to assure the representativeness of the metrics measurements. To address the severity of the proposed adversarial behaviors according to the attacker's available resources, each scenario is simulated where different amounts of malicious peers per CPL, MP, are inserted.

2.7.5 Evaluation Metrics

2.7.5.1 Lookup Success Ratio (LSR)

measures the average ratio of successful lookups over all lookups destined to victim peers. This provides insights about the accuracy of the voting mechanism.

2.7.5.2 Message Complexity (MC)

is the average number of messages exchanged per lookup process until either α replies or i_{max} is reached. This metric is used to provide message overhead calculations for a given lookup.

2.7.5.3 Number of Iterations (NoI)

provides the average number of iterations a given lookup requires to reach α replies which gives an approximation about the average latency of a request.

2.7.5.4 Malicious Detection Rate (MDR)

provides the average number of detected malicious peers per lookup. MDR evaluates the accuracy and scalability of the detection mechanism.

2.7.6 Case Study 1: LEA impact on divPASS using FD behavior

This case study evaluates the impact of LEA using FD-LEA adversarial behavior to highlight the unsuitability of divPASS to mitigate generic LEA in terms of performance and resiliency without the mitigation and the detection mechanisms. Results are evaluated based on LSR, MC, and NoI. As we are evaluating the performance of divPASS under LEA, data is collected only for lookups destined to the victim. We start by describing the experimental results depicted in Figures 2.2 through 2.5, and we close each case study with a detailed interpretation of the results.

Discussion of the results

Figure 2.2a shows LSR of lookups compared to different overlay sizes $N = 5000, 10000, 20000$ and different malicious peers ratios per CPL, i.e., $MP = 5\%, 15\%, 25\%$. As shown, LSR degrades when increasing MP since the probability of intercepting the lookup request by a malicious peer increases. For $MP = 5\%, 15\%, 25\%$, LSR values average between 63% and 91%. This is a significant LSR decrease compared to the divPASS performance in a benign overlay (i.e., $MP = 0$) which results in a LSR between 91% and 100%.

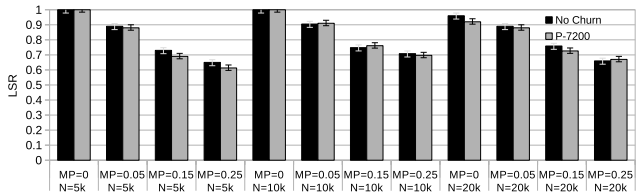
In Figure 2.2b, MC for divPASS average between 7.5 and 11 for different sizes of N and regardless of MP ratio. Figure 2.2c shows NoI results in the range from 1.38 to 1.74 regardless of different choices for N and MP. This means that for a successful lookup, less than two iterations are required to find a peer with an LDE to the victim. Compared to other convergent and divergent algorithms [GRS+14], divPASS provides low latencies as a consequence of the low NoI required for successful lookups.

Interpretation of the results

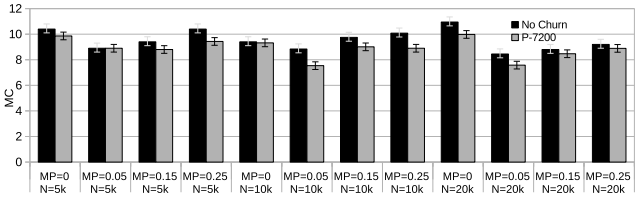
LSR decreases as a consequence of fake destination replies. The reason is that a lookup terminates once a peer replies with an LDE to p_r or when i_{max} is reached; LSR decreases for larger choices of MP.

A major advantage of divPASS is PASS's CPL region choice, such that it tends to resolve peers with LDEs to the destination with high probability. Accordingly, NoI shows low values due to the high probability in contacting a peer that replies with an LDE to the victim. In turn, the number of messages exchanged decreases as only few iterations and peers are contacted until an LDE is found. In addition, terminating the lookup once a peer has sent an LDE reply is a major reason for the low lookup MC.

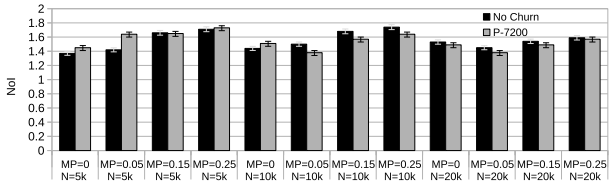
Nevertheless, the results clearly show how divPASS with no additional mitigation and detection mechanisms is susceptible to generic LEA. LSR is severely degraded since lookup results depend only on the first reply. So, we conclude here that although keeping the MC and NoI to minimum is favorable, it imposes a reliability issue for the divPASS algorithm, as shown in Figure 2.2a. To that end, in the next case study we assess our mitigation technique by deviating from the FD-LEA behavior while maintaining a high divPASS LSR and low MC/NoI.



(a) Lookup success rate (LSR)



(b) Message complexity (MC)



(c) Number of iterations (NoI)

Figure 2.2: FD-LEA impact on divPASS using different MP.

2.7.7 Case Study 2: Voting mechanism DMV against LEA attacks

In this case study, we evaluate divPASS's performance after integrating the DMV to evaluate the enhancements of divPASS performance. For better comparison with case study 1, we make use of the same parameter choices for N and MP.

Discussion of the results

LSR results are shown in Figure 2.3a with LSR average values ranging from 83% to 98% which is a remarkable LSR increase compared to case study 1. Figure 2.3b presents MC for divPASS in combination with the integrated DMV. MC during a lookup average between 17.5 and 26 which is relatively higher than

MC values in case study 1 without DMV. As shown in Figure 2.3c, a successful lookup demands NoI between 2.14 and 2.83. NoI results without mitigation were lower in the first case study, where values averaged between 1.38 to 1.74.

Interpretation of the results

The remarkable enhancement in the LSR values is due to integrating the mitigation mechanism through the DMV. Basically, due to the assessing criteria of the voter, chances of picking a correct reply is considerably high even in case where $MP = 25\%$. We note that, increasing MP escalates the probability of malicious peers to get picked and thus, the number of fake replies increases which in turn degrades the system's reliability.

The noteworthy MC increase compared to case study 1 is due to the fact that the voter maintains the lookup process until α replies are received, which is not the case in case study 1 where the lookup terminates once the first resolving reply is received. Due to divPASS's approach to query only a certain CPL range which expectedly contains a high percentage of peers with LDEs to the destination, the NoI required to receive α replies are very small, i.e., between 2 to 3 as observed from Figure 2.3c. As a result, due to low NoI needed to reach α replies, MC ranges provide an acceptable increase compared to case study 1 where only a single reply is required.

From the results, we assert that divPASS algorithm combined with the DMV provides a very good performance in mitigating LEAs as it provides high LSR values while keeping MC and NoI minimized compared to case study 1.

Our mitigation model for high LEAs shows that divPASS, in conjunction with the proposed mitigation mechanism, is scalable to maintain overlays with thousands-millions of peers.

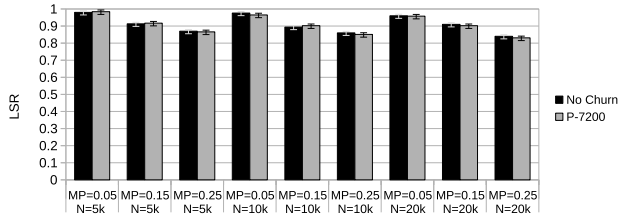
2.7.8 Case Study 3: LEA impact using weighted FD-PS impact

In this case study, we assess the impact of launching a LEA using weighted FD-PS to show the effect of the proposed attack behavior combinations on divPASS based lookups.

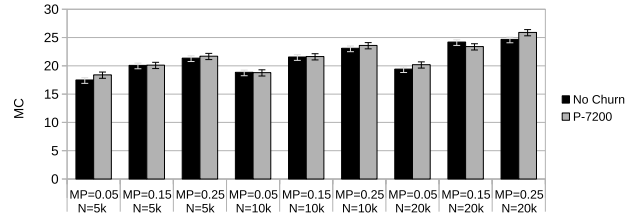
Here FD and PS are weighted equally, i.e., the probability that a malicious peer will choose an FD-LEA or a PS-LEA behavior is 0.5. Same overlay sizes $N = 5000, 10000, 20000$ and same ratios of malicious peers per CPL, where $MP = 5\%, 15\%, 25\%$ are used in these experiments. We note that scales for comparable figures may vary due to distant ranges of values as can be seen in NoI values between Figure 2.4c and Figure 2.5c.

Discussion of the results

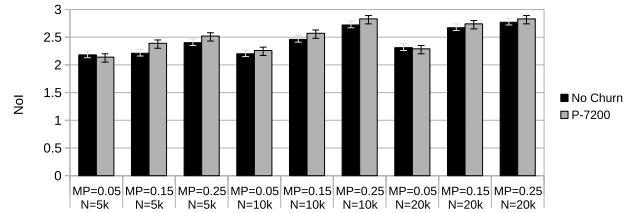
Figure 2.4a shows the LSR for different overlay sizes and MP ratios. Values range between 63% and 94%. Obviously, when MP increases, the LSR value decreases accordingly as more malicious peers are able to intercept the parallel lookup requests sent by p_r. In Figure 2.4b, we notice that a remarkable MC increase stems from the message exchange until α replies are received; MC values range between 19 and 34 messages. Figure 2.4c shows an increase for NoI, i.e.,



(a) Lookup success rate (LSR)



(b) Message complexity (MC)



(c) Number of iterations (NoI)

Figure 2.3: Combined divPASS/DMV performance with different values for MP.

values range from 2.2 to 4. Compared to case study 2 where only FD-LEA are launched, the NoI increased 30% in the weighted FD-PS LEA.

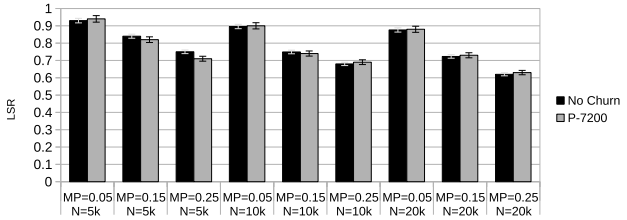
2.7.8.1 Interpretation of the results

The noticed LSR decrease occurs due to the combined effect of both FD-LEA and PS-LEA behaviors which can be summarized as follows: (i) the impact of fake replies that are sent to the voter and (ii) the increment of malicious peers' ratio due to PS-LEA effect where malicious peers intentionally insert more malicious entities into p_r's candidate list. Moreover, due to (ii), NoI required for a successful lookup increases as the number of malicious peers inside the candidate selection list increases. As a result, the probability of picking more malicious peers for the next rounds increases, which in turn forces the divPASS algorithm to run more iterations until α replies are received.

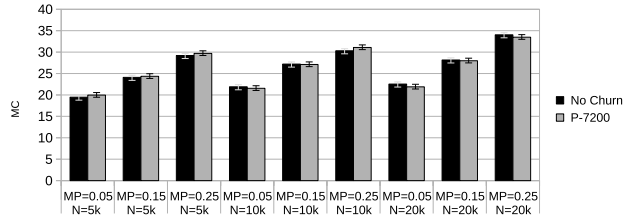
Increasing the NoI has an impact on the average number of messages exchanged during a lookup as it requires contacting more peers. Accordingly, MC increases per lookup. To that end, we conclude the unsuitability of divPASS with no detection mechanism to mitigate generic LEAs.

We note that according to our observations from running experiments for different weights, Increasing the weight of PS behavior have a direct impact

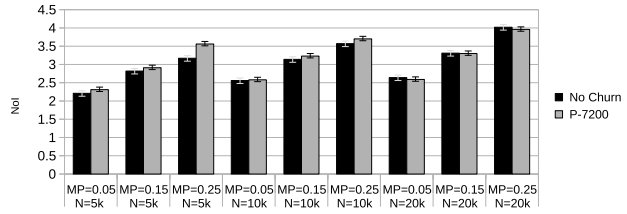
on the average NoI and MC which is the target of PS attacks. Meanwhile, increasing the weight of FD-LEA behavior impacts negatively on the LSR values. For instance, running the same experiment with an FD-LEA weight of 0.25 and a PS-LEA weight of 0.75, we achieve results with MC= 47 and NoI= 5.37.



(a) Lookup success rate (LSR)



(b) Message complexity (MC)



(c) Number of iterations (NoI)

Figure 2.4: Baseline results for FD-PS without detection.

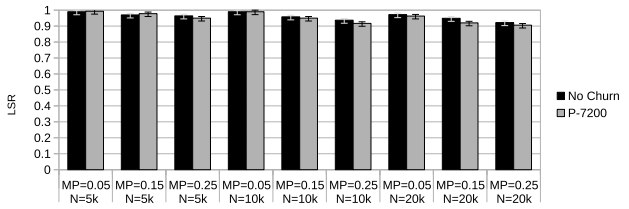
2.7.9 Case Study 4: Detection and mitigation mechanisms response against weighted FD-PS behaviors

Now, we assess the performance of divPASS when integrating the proposed mitigation and detection techniques against FD-PS LEAs. We evaluate the results compared to the previous case study where no detection mechanism were integrated.

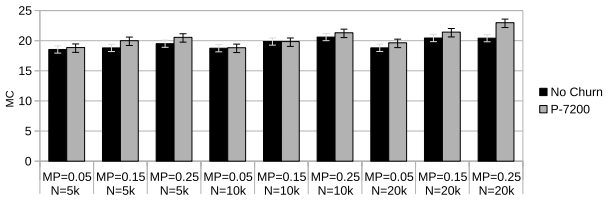
Discussion of the results

In Figure 2.5a, LSR values average between 90% and 99% which is a noticeable increase compared to case study 3, even in scenarios where $MP = 25\%$, where LSR averaged between 63% and 94%. Figure 2.5b shows the average MC where values range between 18.5 and 23. In this scenario, average MC are relatively

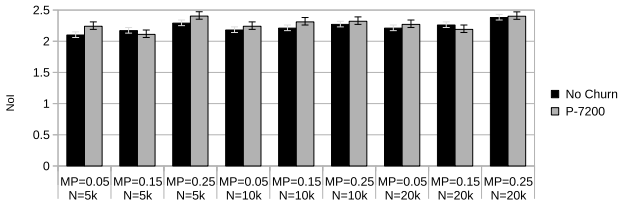
similar to case study 2, where only FD-LEA based attack is running. Moreover, in Figure 2.5c, average NoI average between 2.1 and 2.4 which is again relatively similar to case study 2 where no malicious peers launched a PS-LEA. A noticeable decrease is noted, comparing NoI and MC values to case study 3. Since we aim to evaluate the efficiency of our detection and mitigation mechanism, we evaluate the MDR per lookup. In Figure 2.5d, MDR averaged between 0.55 and 6 depending on N and MP which provide insights about the average number of malicious peers that launch a PS-LEA contacted during a lookup.



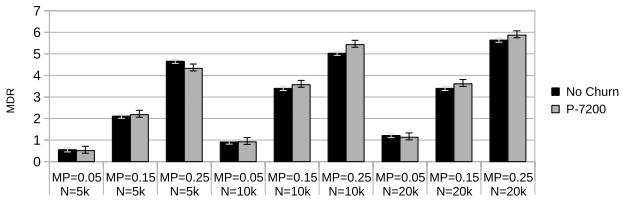
(a) Lookup success rate (LSR)



(b) Message complexity (MC)



(c) Number of iterations (NoI)



(d) Malicious detection rate (MDP)

Figure 2.5: Detection performance of FD-PS based LEAs.

2.7.9.1 Interpretation of the results

After DMV integration with divPASS, a remarkable LSR enhancement can be noticed. This is due to the fact that detecting malicious peers before insert-

ing malicious entities decreases the probability of polluting the candidate list which in turn reduces the probability of launching FD-LEA by other malicious peers. Moreover, once a peer is detected, the MP value decreases which in turn enhances the chance of p_r to contact benign peers. For the same reason, NoI decreases since α replies can be collected in less number of iterations. Accordingly, the number of messages that needs to be exchanged during a lookup decreases.

For high values of MP, the detector shows that malicious peers manifesting a PS-LEA behavior contacted during a lookup call are detected. In fact, MDR values underline the detector's positive impact on LSR, NoI, and MC.

We conclude that the combination of our detection and mitigation mechanisms yields excellent reliability and performance in the presence of FD-PS LEAs. Also, it is a preparatory step to achieve reliable and decentralized malicious peer removal from overlays.

2.8 Conclusion

We briefly conclude this chapter in the following. However, in Chapter 7, where the main conclusion of the whole thesis is presented, the main contributions related to Research Question 1 are depicted.

Localized Eclipse attacks (LEA) pose a significant threat to P2P-based applications. We extend the divergent lookup mechanism, which was originally developed to mitigate the specialized topology-aware Eclipse Attack (taLEA), to mitigate the more generic LEA.

Moreover, we introduced a sophisticated attacker model which causes significant decreases in reliability and performance in divergent lookups. Consequently, we integrated a new detection mechanism that allows identifying attack peers with high accuracy. We, furthermore, assessed the performance of our novel eviction mechanism, which is the second sub-contribution Chapter 3 that we describe in detail in the following chapter.

Chapter 3

Structured Overlays: Evicting Malicious Peers

In this chapter, we address Research Question 1 and the second sub-contribution ***(C1.2)***. In a summary, this chapter presents the work done towards designing a light-weight eviction mechanism that builds upon the detection mechanism described in Chapter 2 to counter a progressive variant of LEA attacks. This work was published in [IGS16]. Note that the system model already described in Section 2.3 holds and is used as is throughout this work.

3.1 The Necessity of Eviction

For some P2P applications, a subset of peers might experience request rates above average by both, legitimate users as well as attackers. This can be due to the criticality or the popularity of their stored data, centralized services such as authentication hosted by that particular subset, or their considerable storage/bandwidth provision. In particular, due to the delay intolerance and fast response time constraints for applications such as video streaming or online gaming, a subset of peers is assigned more responsibilities to support the required QoS such as high level peers in tree based P2P streaming [LLR09], or peers promoted as super peers [YK13]. Such peer subsets represent vital assets that ensure reliable overlay service provision. Unfortunately, various "Localized Attacks" (LA) target specific peer subsets. LAs continuously evolve and can severely degrade an overlay's reliability and performance due to the perturbations they cause using only a comparably small amount of malicious peers. Examples for LAs are Eclipse attacks (EA), sybil, index poisoning and DDoS attacks [SCD+04; DH06; KCW10; LNR06; ZJT13; STR10b; KLK+12].

In our previous work [GRS+14; IGS15; GIS15], we proposed an effective mitigation and detection technique that addresses LAs. In addition, a variety of mitigation techniques have been proposed to counter LAs [SND+06; CCF+12; GIS15; LYL14; FMR+09].

However, the existing mitigation techniques lack generic applicability, as their efficiency is either bound to specific P2P networks or LA variants. Also, to the best of our knowledge, no mitigation technique exists that is, generally applicable for a wide range of LA variants, detects malicious peers, and announces

the eviction of previously detected malicious peers to benign ones [CCF+12; BM07; LML13].

The absence of an eviction mechanism leaves overlays exposed to severe security threats since malicious peers are only known to a small fraction of peers. Consequently, malicious peers can effectively exploit benign peers' partial view of the overlay, e.g., by targeting newly joined peers or peers which are inept to detect their malicious behavior. Malicious peers can adapt to new adversarial variants to overcome particularly fitted detection techniques. Detection techniques that are only locally executed by individual peers may be rendered useless from an overlay service perspective.

For the aforementioned reasons, we emphasize the necessity of proliferating information about detected malicious peers to the majority of peers in the overlay as a precondition for their eviction. To that end, we propose a novel two-fold eviction mechanism based on the formation of distributed quorums which reliably propagate information about malicious peers.

Contributions:

We focus on the following aspects in this work:

- Assessment of a distributed eviction mechanism that is (i) generally applicable to various LA types (ii) able to effectively evict malicious peers from the overlay and thus, restore reliability and performance requirements, and (iii) capable of propagating the existence of malicious peers to the rest of the overlay.
- Development of an LA model that does not assume a specific LA instance and thereby offers generality and extensibility, and provides the required parameter landscape for evaluation of the eviction mechanism.

The evaluation of the proposed mechanism shows high detection and eviction rates for sophisticated LA variants of up to 99% for up to 10% malicious peers in overlay networks of varied sizes.

In Section 3.2, we discuss the related work addressing the existing LAs and the proposed mitigation techniques. The proposed LA model is discussed in Section 3.3. Next, the technical details of the proposed eviction mechanism are provided in Section 3.4. Finally, we evaluate the impact of the proposed LA and the effectiveness of the eviction mechanism in Section 3.5.

3.2 Related Work

The severity of LAs along with various countermeasures has been addressed in literature. Drawbacks of existing techniques include the limited applicability to specific LA variants, the need for centralized coordinating peers, sophisticated encryption schemes, or their lack of an eviction mechanism.

A mitigation and detection mechanism is proposed in [CCF10] that focuses on removing suspicious peers from the list of possible candidates to contact. However, the mechanism does not address the eviction of malicious peers from benign peers Routing Table (RT). In [CCF+12], the authors introduce a network crawler for KAD, a Kademlia based network, for monitoring and detecting malicious peers. However, no eviction technique is introduced.

A detection mechanism for streaming P2P applications using a belief propagation algorithm is introduced in [GG13]. In [FMR+09], a study on the severity of EAs on KAD is conducted with proposing a mitigation technique based on a trusted cryptographic scheme. Nevertheless, in both studies, no eviction is proposed in addition to the usage of a centralized approach.

In [BM07], the authors introduce a mitigation mechanism for LAs based on assuring multiple, disjoint paths during lookup initiation. However, the proposed mitigation mechanism uses a cryptographic scheme and no malicious removal mechanism is proposed. Similarly, a self-eviction scheme against false routing information is proposed in [LML13]. In addition to the absence of a propagating criterion for detected malicious peers, the mechanism is based on strong encrypting technology.

In [LL10], a stochastic detection and removal scheme is proposed as a countermeasure against pollution attacks. However, the mechanism is only applicable for pollution attacks and in P2P streaming systems. Furthermore, a mitigation scheme against DDoS attacks via validating membership information is introduced in [STR10a]. Nonetheless, no eviction mechanism was introduced. In [ST15], the authors propose a detection scheme against sybil attacks by calculating trust values for each peer joining the overlay. However, the scheme relies on central entities, specific to a single LA variant and no evaluation is provided.

The authors in [LMS+10] highlight the basis of conducting the most commonly launched LAs, such as EAs, publish attack and node insertion attack. The common aspect while launching the aforementioned LA variants is intercepting messages destined to the victim via poisoning benign peers RT. However, the paper proposes no mitigation or eviction mechanism.

Next, as discussed in [LMS+10] we address a generalized LA model that constitutes the fundamentals of various existing LAs for evaluating our proposed eviction mechanism.

3.3 Localized Attack Model

This section presents the attack model. It is the basis to assess the central contribution of this particular work in the thesis, i.e., the eviction mechanism, which will be presented afterwards.

The novelty of this attack model is its generality as it covers a wide range of existing LAs, e.g., EA, sybil, poisoning attacks and DDoS. Hence, the resilience of the proposed eviction mechanism is validated for a diverse set of LAs.

As discussed in Section 3.2, the severity of LAs correlates with the amount of lookup messages that are intercepted by malicious peers and which are meant to resolve the contact information of the victim peer p_v. Our attack model focuses on different adversarial strategies and behaviors that illustrate the trade-off between immediate attack severity and detection hardness. In our model, the amount of malicious peers and their placement in the overlay are referred to as strategies, whereas the behavior refers to the interaction of malicious peers with benign ones that deviates from the specification of the P2P network's protocol. The next three subsections describe the strategies and the behaviors.

3.3.1 Malicious Resources

Although using a large amount of malicious peers might increase the LA's severity, this has also drawbacks in terms of an increased detection probability, as well as higher LA cost.

Recent LA studies [CCF+12; IGS15; GIS15] indicate that malicious insertions of only 5-10% in terms of the overlay size is sufficient to intercept a large majority of lookups requesting p_v's contact information. Therefore, we focus on that percentage range for malicious insertions in the overlay.

3.3.2 Malicious Placement

Now, we discuss the strategy for placing malicious peers to maximize lookup interception. As presented in Section 2.3, various lookup mechanisms may be used by P2P networks.

Depending on the particular lookup mechanism in a P2P network, patterns on the lookup request message forwarding among benign peers that try to resolve the contact information of p_v can be determined by the attacker. Hence, an efficient placement focuses on overlay regions that reveal a higher probability of receiving such lookup requests.

In proposed in [GIS15; GRS+14], a divergent lookup mechanism is considered that spans the whole address space, i.e., lookups are equally probable of being forwarded to any region in the address space. Here, we make use of *divergent Random Walks* mechanism which is based on random peers selection while restricting only peers within p_v's proximity for forwarding lookup requests. This means that placing malicious peers uniformly across the address space yields equal probability that malicious peers, independent of their location in the address space, intercept lookup requests destined to p_v. Such placement provide a full overview about the generality and suitability of the proposed EM.

Once a malicious peer has been placed, it can launch an LA by performing different adversarial behaviors, which are discussed in the next subsection.

3.3.3 Adversarial Behaviors

This subsection highlights various adversarial behaviors of malicious peers as a mean to intercept lookup requests addressed to the victim p_v. Lookup message intercepting LAs follow a threefold approach: (i) poison benign peers' RTs, (ii) malicious collusion, (iii) dynamically alternate adversarial behavior. Each behavior is described below.

3.3.3.1 Poisoning benign peers

Depending only on the inserted malicious peers to intercept lookups destined to p_v is suboptimal due to the limited amount of malicious peers. In order to let benign peers unknowingly partake in the LA execution and promote the interception of lookup messages by malicious peers, benign peers RTs are poisoned, i.e., RT entries pointing to p_v are altered to point towards a malicious peer.

Initially, malicious peers propagate a fake reply regarding p_v by (i) pretending to own p_v's contact information, (ii) advertising the contact information of another malicious peer as the destination contact information. Fake replies are a common practice to achieve RT poisoning.

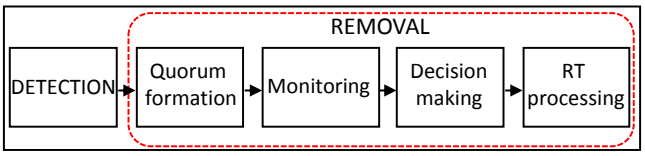


Figure 3.1: Eviction process overview

Once a peer with a poisoned entry towards p_v receives a lookup request, it replies with poisoned information that points to a malicious peer, hence, the lookup initiator forwards the lookup request to. Such adversarial behavior is shown to yield severe impact on the overlay as substantiated in Section 3.5.

3.3.3.2 Malicious collusion

In order to evaluate the performance of the proposed eviction mechanism in the worst case LA scenarios, we assume that malicious peers are capable of colluding through informing each other about their status. This means that each malicious peer has a periodically updated list about the location and the status (on-line, detected, removed) of other malicious peers.

Moreover, malicious peers are able to exchange messages to inform each other once a peer is detected. As a consequence, malicious peers can alternate between sending fake or correct replies in order to falsify monitoring procedures used by the detection and eviction mechanisms.

3.3.3.3 Alternating behavior

The probability of generating a fake reply is controlled through a configurable parameter for the attacker. We refer to this parameter as FR. Once p_m intercepts a lookup request destined to p_v, p_m sends a fake reply with probability FR to p_i.

In Section 3.5, we evaluate how FR is a vital parameter for the attacker in terms of detectability and LA severity.

3.4 Eviction Mechanism

In this section, we describe the technical aspects of the Eviction Mechanism (EM) proposed as a countermeasure against general forms of sophisticated LAs. In order to effectively evict the overlay from malicious peers, an accurate detection scheme must be applied beforehand that allows to detect peers that exhibit possible malicious behavior.

As depicted in Figure 3.1, the EM is divided into two main blocks, *Detection* and *Removal*. We start by illustrating the detection process that allows peers to locally suspect certain peers based on their lookup replies. Afterwards, the removal process which is responsible for inspecting suspected peers and consequently evict malicious peers is described.

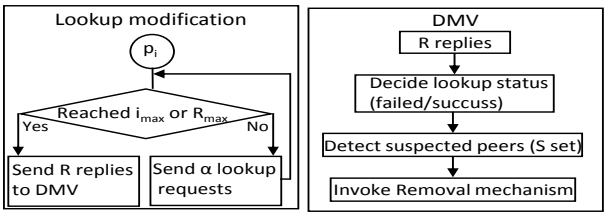


Figure 3.2: Detection procedures

3.4.1 Detection Process

The objective of the detection process is to enable peers to locally suspect a given peer according to its lookup reply.

In previous work [IGS15], we introduced a modified lookup mechanism which allows peers to gather more than a single lookup reply in order to compare replies according to (i) the average number of hops for each reply compared to the known average from previous lookups, (ii) the compliance of the replying peers' location with the lookup protocol and (iii) the destination contact information returned in the replies. Once a peer violates any of these detection criteria, this peer is announced to be suspected. The original implementation of the P2P lookup mechanism is based on accepting the first reply that contains the requested information about a given peer p_v. This coerces the lookup initiator p_i to accept the lookup result without being able to validate the results since only a single reply is considered. In turn, whenever a malicious peer receives the lookup request and generates a fake reply, p_i consequently accepts the reply which poisons p_i's RT.

The developed modified lookup mechanism which allows p_i to consider a set of replies instead of only the first received reply, is discussed below.

3.4.1.1 Lookup modification

From the original operation of the lookup mechanism, i.e., before introducing the modification, p_i picks α candidate peers from its RT to start forwarding a lookup request for p_v's contact information. Once peer p_r receives such a request, it replies with p_v's contact information if p_r has an entry for p_v in its RT. Otherwise, p_r replies with a list of peers that, according to the routing protocol, have a high chance of knowing p_v. p_i generates α new requests from this list for the next iteration. This process terminates immediately once p_v's address is resolved or i_{max} iterations are reached.

Our lookup modification continues the lookup process until R replies containing p_v's contact information are received or i_{max} iterations are reached, where $R \leq \alpha$. Consequently, p_i will be able to compare multiple replies from different peers and, thus, decide whether certain peers are suspicious.

For this purpose, we make use of a Dynamic Majority Voter [BJ91] (DMV) to process the R received replies. The details of the DMV are discussed below.

3.4.1.2 DMV

From the modified lookup mechanism, peers are compared according to their replies given the aspects described earlier and suspected peers are added to the suspicion set S, where $S \subset R$. Afterwards, the remaining unsuspected peers

from the set of R are provided as inputs to the DMV. Note that a suspected reply in this context refers to a suspected peer as replies are only accepted from distinguished peers, i.e., no peer can provide more than a single reply.

The DMV decides whether a valid majority of identical, non-empty replies exists or not. If such majority is found, the lookup reply is considered successful and p_i stores the corresponding contact information. Otherwise, the lookup is declared to be unsuccessful and consequently, p_i initiates a new lookup to resolve p_v's contact information.

In addition, the unmatched minority is added to the suspicious list, i.e., the remaining set of replies that did not constitute the majority. Detailed technical description about the detection process is available in previous work [IGS15].

Once the detection process announces a set of suspected peers, the removal process is invoked to further inspect and accordingly evict malicious peers. Next, the technical concepts of the removal process are described.

3.4.2 Removal Process

The basic functionality of the removal process is to further inspect suspected peers and evict peers confirmed to be malicious. In order to do so, a distributed process is required to monitor the suspected peers and then reach a decision about their status. Afterwards, peers that turn out malicious are evicted from peers RT.

The removal process comprises four main procedures. First, *Quorum Formation* defines the criteria of forming a distributed quorum. Second, *Monitoring* handles monitoring the behavior of suspected peers. Third, *Decision Making* is responsible for reaching a decision regarding each suspected peer. Finally, *RT processing* defines the removal criteria about peers confirmed to be malicious.

For more convenience, a list of all abbreviations used within the following sections is provided in Table 3.1.

Table 3.1: LAs eviction: acronyms description

Var.	Description	Var.	Description
p_s	Suspected peer	S	Set of suspected peers
p_i	Quorum initiator	p_v	Victim peer
Q	Set of quorum peers	p_q	Peer joined Q
R	number of replied peers	FR	fake reply probability

3.4.2.1 Quorum Formation

Here, we describe how the initiating peer p_i forms a quorum Q, as depicted in Figure 3.3a. Each process is defined in terms of mechanism and interpretation about the notions behind the development of each process.

Mechanism

Once p_i detects suspicious peers $p_s \in S$ through the DMV, p_i executes the following steps:

(a) p_i sends a lookup notification to all R peers, recall that R peers contain the set of all peers that replied with p_v's contact information to p_i. A lookup notification contains an acknowledgment that some replies are suspected. The notification message does not convey any information about the identity of the suspected peers, however, its purpose is to alert benign peers about the possibility that they have replied with poisoned entries.

(b) p_i selects only unsuspected peers from the R peers to form a quorum. The set of peers that form a quorum is referred to as Q, including p_i.

(c) p_i sends a quorum joining request to each peer $p_q \in Q$. The joining request contains: (i) a list containing all suspected peers and (ii) a time-stamp that defines when p_q has to start monitoring each p_s.

(d) Each p_q replies to p_i with either an acceptance or rejection to the quorum joining request. p_q may reject a Q_R due to several reasons as p_q might be malicious, already joining another quorum, due to low CPU capabilities or currently experiencing loaded network traffic.

Process interpretation

Malicious peers exist in Q may provide correct replies according to the FR parameter discussed in Section 3.3.3. Therefore, colluding malicious peers are capable of informing malicious peers about being suspected.

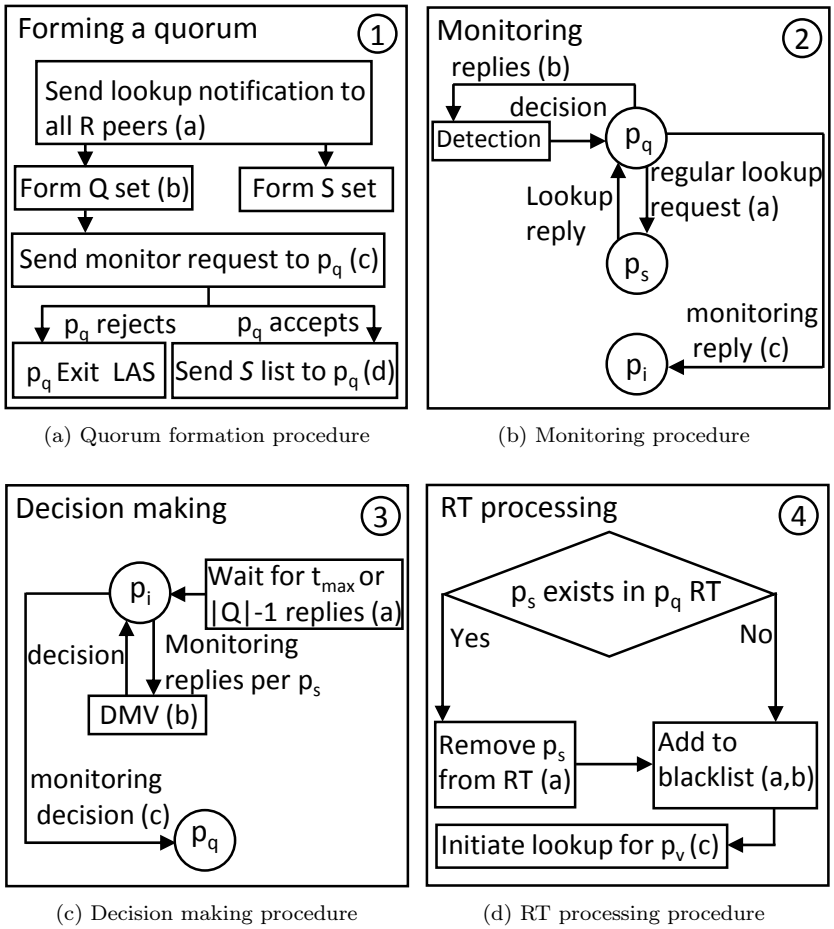
As a countermeasure to restrain suspected malicious peers from changing their behavior if they reveal the identity of any p_q, p_q receives only a list of suspected peers without being informed about other peers in Q or their time-stamps. Malicious peers that are informed about being suspected can behave as follows:

1. Decrease the FR parameter, i.e., behave benignly to avoid being evicted if confirmed malicious.

2. Keep providing fake replies with the same rate, i.e, exploit chances to poison benign peers RT regardless of the risk of being evicted.

Poisoned peers exist in S due to providing suspicious replies to the DMV. For this reason, poisoned peers make use of the notification message to restore their benign state. Once a poisoned peer receives a notification message, it initiates a lookup requesting p_v's contact information in order to re-evaluate its reply to p_i. After a poisoned peer proves its benign state, it receives the suspected set S without joining Q, i.e., it can monitor peers in S and accordingly removes malicious peers based only on its own decision. The main advantage is that poisoned peers are able to restore p_v's correct contact information in addition to removing other malicious entries that might exist in their RT, which allows for higher removal rate of malicious peers.

Churning peers in Q refer to either benign or malicious peers that might leave the overlay or do not complete the removal procedures. Due to the small number of peers that forms a quorum, procedures are executed over a short time frame. Hence, the churning amount of peers do not interrupt the removal process or deviate the final decision of the quorum as evaluated in Section 3.5.



(a) Quorum formation procedure

(b) Monitoring procedure

(c) Decision making procedure

(d) RT processing procedure

Figure 3.3: Removal Procedures

3.4.2.2 Monitoring procedure

This procedure defines the monitoring basis that p_q follows with each p_s and how results are gathered at p_i, as shown in Figure 3.3b.

Mechanism

Once p_q accepts joining quorum Q, the following steps are executed in parallel $|S|$ times by p_q.

(a) p_q sends a regular lookup request, using the modified lookup mechanism, to each $p_s \in S$ requesting p_v's contact information according to the time-stamp assigned by p_i in the quorum joining request. Note that each p_q sends a single lookup request to p_s, however, in order to be able to decide about p_s, it sends a lookup request to other $\alpha - 1$ peers, i.e., modified lookup mechanism.

(b) After p_q processes the received replies via the DMV, p_q decides accordingly whether p_s is malicious or not.

(c) p_q sends a monitoring reply to p_i containing the DMV decision about each p_s, i.e., a boolean that depicts the monitoring result.

Process interpretation

The purpose of using regular lookups is to provide anonymity to the monitoring procedure so as malicious peers can not differentiate between lookups intercepted from normal peers and those intercepted from p_q. Moreover, p_i assigns different time-stamps to each p_q so that p_s does not receive multiple lookup requests simultaneously more than its expected in-going bound to assure p_s does not detect any abnormal behavior.

Through the detection mechanism, p_i uses the DMV to process lookup replies that include p_s's reply, and hence, decide whether p_s is malicious or not. Notably, if the DMV does not suspect p_s, this might be inferred that p_s was in a poisonous state.

3.4.2.3 Decision making procedure

In this procedure, we discuss the procedure that allows to reach a decision about p_s, as illustrated in Figure 3.3c.

Mechanism

p_i reaches a decision about p_s through executing the following steps:

(a) p_i waits for either timeout t_{max} or receives $|Q|-1$ monitoring replies.

(b) p_i inputs the received monitoring replies about each $p_s \in S$ separately to its DMV.

(c) Finally, p_i sends the DMV results to each $p_q \in Q$, we refer to this message as "monitoring decision reply".

Process interpretation

Timeout t_{max} assures malicious peers do not block the removal process since a malicious peer in Q might not send a monitoring reply to p_i. t_{max} is set according to the latest time-stamp assigned to any peer in Q.

The monitoring decision message includes p_i's decision about each p_s, i.e., p_s is malicious or was poisoned with malicious entry. Consequently, each $p_q \in Q$, proceeds to the "RT processing" procedure.

3.4.2.4 RT processing

Here we define the criteria of removing malicious peers from RTs as depicted in Figure 3.3d. Moreover, we highlight how information about malicious peers are propagated through the overlay.

Mechanism

After p_q compares the received decision from p_i about each p_s with its own decision, the following steps are executed when p_q decides to proceed with removing p_s. Note that different comparison cases are discussed in details in the procedure interpretation.

(a) In case p_s exists in p_q's RT, p_q removes p_s and adds it to a blacklist to assure no further contact with p_s.

(b) Otherwise, in case p_s does not exist in p_q's RT, p_s is added to p_q's blacklist to hinder adding p_s in its RT in the future.

(c) p_q initiates a new lookup for p_v's contact information. During different lookup iterations, no replies or candidates suggestions will be accepted if such peers exist in p_q's blacklist.

Process interpretation

p_q compares the monitoring decision received from p_i about each p_s with its own decision about p_s concluded during the monitoring procedure, i.e., p_q checks whether both decisions match or not.

Recall that p_q's possible decisions about a suspected peer are: 1) Benign, 2) Suspicious. Similarly, for p_i, possible states reported in the monitoring decision reply are: 1) Poisoned, 2) Malicious. For simplicity, we state here the consequent decisions according to the possible combinations.

p_i: Poisoned, p_q: Benign

In this case, p_i states that p_s was poisoned, which confirms p_q's status that p_s is benign. As a result, no further action is taken about p_s.

p_i: Poisoned, p_q: Suspicious

This denotes that either p_s was poisoned during the monitoring period of p_q, or p_s is malicious but the majority of monitoring replies was correct due to low FR at p_s. At this point, p_q initiates new lookup request to p_s asking for p_v's

contact information to decide whether to remove p_s or not before deciding to initiate a quorum to monitor p_i.

p_i: Malicious, p_q: Benign

This case refers to p_s being malicious according to the majority of monitoring replies reported to p_i. However, due to low FR, p_q reached to a decision that p_s is benign. p_q initiates a new lookup in order to re-monitor p_s before deciding to start monitoring p_i. In fact such approach assures malicious peers do not exploit the removal procedure. Meanwhile, p_i further proceeds to the removal steps.

p_i: Malicious, p_q: Suspicious

As both p_i and p_q confirms p_s being malicious, p_i and p_q proceed to the removal steps.

3.5 Evaluation

In this section, we provide an evaluation of the proposed EM against general attack model that constitutes the basis of various specific LAs. We first start with case study 1 "LA impact" that evaluates the impact of launching a severe LA on a P2P overlay. Case study 2 "EM evaluation" assesses the performance and effectiveness of EM.

First of all, the simulation environment, parameters and metrics used for evaluation are introduced. Afterwards, each case study is presented with results discussion and interpretation. Finally, a summary that highlights the main results and conclusion about EM is provided.

3.5.1 Simulation environment

Case studies were conducted using the OMNeT++ simulator [Pon93] and OverSim [BHK07] which provides various P2P protocol implementations. Each simulation experiment was running for 4 hours. Moreover, for confidence interval measurements, each simulation was scheduled for 10 repetitions. In Table 3.2, the simulation parameters used in the experiments are provided.

Table 3.2: LAs eviction simulation parameters

Parameter	Value
Maximum iterations (i_{max})	10
Number of victims ($\|V\|$)	1
Maximum received lookup replies (α)	9
Key length (w)	128
Lookup	Divergent Random Walks [GRS+14]
Malicious Insertion ratio (MI)	$5\%, 10\%$
Overlay size (N)	$5k, 10k, 20k, 30k$
Fake Reply probability (FR)	$50\%, 80\%$

3.5.2 Simulation model

In order to validate the scalability of our approach, EM is assessed using different overlay sizes, i.e., $N = 5k, 10k, 20k, 30k$. Different malicious insertion ratios $MI = 5\%, 10\%$ and fake reply probabilities $FR = 50\%, 80\%$ are used to represent the impact of varying amounts of malicious peers with different probabilities of generating fake replies, resulting in 16 different overlay configurations.

3.5.2.1 System workload

Our target is to base our evaluation on launching an LA on P2P networks with special set of peers that are more frequently contacted and offer special services to the overlay. For this reason, simulations are based on a "Service Overlay Network" where 80% of lookup requests are addressed to the victim. In general, lookups are sent on average every 10 seconds with 5 seconds standard deviation.

3.5.2.2 Simulation Churn Models

In order to simulate churning rate of peers, a **Pareto (P-500)** is used where the average life-time and dead-time of peers is 500 seconds. The choice of such distribution is due to the realistic experimental results provided for P2P overlays in [ZL06].

3.5.3 Evaluation Metrics

3.5.3.1 Lookup Success Ratio (LSR)

the ratio of successful lookups to the total number of lookups initiated to the victim only. LSR assesses the reliability of the network.

3.5.3.2 Message Complexity (MC)

the overhead exerted on the system due to lookups initiation, malicious existence and EM procedures execution.

3.5.3.3 Poisoned Replies (PR)

the average number of poisoned replies per lookup. This metric is used to assess the impact of poisoning benign peers RT on the victim's service provision.

3.5.3.4 Malicious ratio per RT (MRT)

the average ratio of malicious entries in benign peers RT. This metric evaluates the impact of malicious peers insertions.

3.5.4 Case Study 1: LA Impact

In this study, we assess the impact of launching an LA with the proposed adversarial behaviors discussed in Section 3.3 and how poisoning benign peers RT can severely degrade the system's reliability.

Discussion

As shown in Figure 4.4a, LSR shows negligible values that average below 1% due to LA impact, i.e., more than 99% of lookups initiated to resolve p_v's contact information fail. MC overhead is depicted in Figure 4.4c where values are in the range of 8 to 13. For $MI = 10\%$, MC is slightly lower, 8-9 messages, compared to values observed for $MI = 5\%$.

In Figure 3.4c, the average number of poisonous replies per lookup is remarkably high as ranges are between 78% and 90%. PR values For $MI = 5\%$ are higher than in $MI = 10\%$ with an average of about 6%. Finally, Figure 4.4b depicts the average MRT. For $MI = 5\%$, values ranges between 20%-22%. For $MI = 10\%$, higher existence of malicious peers is observed in peers RT where values average between 25%-26%.

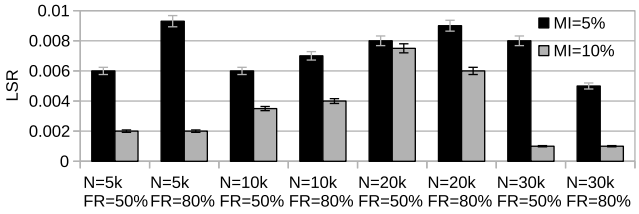
Interpreting the results

Due to the malicious existence and poisoned peers which propagate malicious information, lookup requests are almost completely intercepted as depicted in Figure 4.4a. Consequently, p_v's service provision is markedly degraded as 99% of lookups initiated to p_v fail.

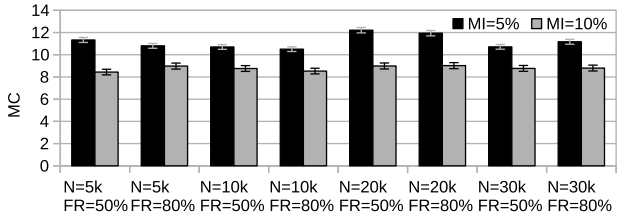
From [GRS+14], the average MC overhead using divergent Random Walk mechanism is in the range of 11-13. However, as indicated for particular LA configurations, MC indicates less overhead as indicated in Figure 4.4c. The reason is that benign peers' RT entries pointing to p_v are poisoned with malicious peers that fake storing p_v's contact information. Accordingly, such peers reply with malicious information and the whole lookup is *falsely* resolved in the first or second iteration at most which lowers MC value. For the same reason, MC is lower for $MI = 10\%$ as more malicious peers intercept the lookup request than for $MI = 5\%$ and thus, reply with fake replies which accelerates collecting α replies which terminates the lookup process. In fact, such abnormal MC is one of the criteria used by the detection mechanism to suspect malicious peers.

PR highlights the impact of poisoning benign peers entries towards p_v as an application of the adversarial behavior of malicious peers as discussed in Section 3.3. As shown in Figure 3.4c, a smaller amount of malicious peers as $MI = 5\%$ yields a higher probability for poisoned peers to receive lookup requests than for $MI = 10\%$ where a larger amount of malicious peers intercept the lookup request. Regardless of the selected value of FR, a large fraction of benign peers is poisoned due to the severe impact caused by small MI. Such a large fraction of poisoned replies per lookup is the main reason of the resulting severe degradation in LSR.

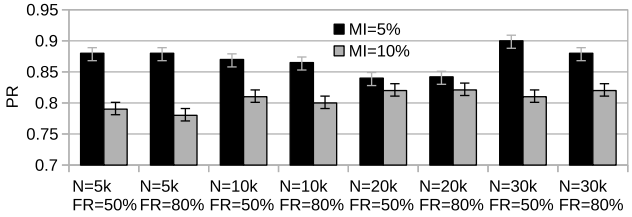
As illustrated in Figure 4.4b, malicious peer insertions in the range of 5%-10% is capable of polluting 20%-26% of benign peers RT. The reason for that is the propagation of entries pointing to malicious peers. Consequently, whenever a peer selects α different peers from its RT for the first lookup iteration, or replies to a lookup request with a list of possible candidates, malicious peers are selected with high probability. The severity of launching LA various forms can be concluded from the small amount of required malicious insertion to completely intercept messages destined to the victim.



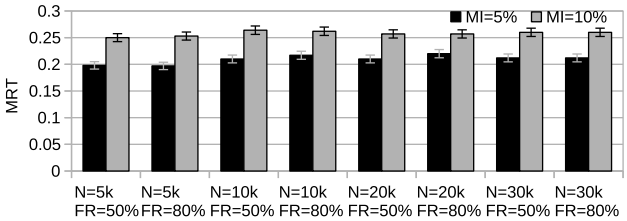
(a) Lookup success rate (LSR)



(b) Message overhead complexity (MC)



(c) Poisoned Replies per lookup (PR)



(d) Malicious ratio per RT (MRT)

Figure 3.4: LA impact

3.5.5 Case Study 2: EM Evaluation

Now, we evaluate the performance of the proposed EM in terms of the detection and propagation effectiveness. For comparability reasons, the same set of metrics and evaluation criteria are used.

Results discussion

As depicted in Figure 4.5a, activation of EM results in high LSR rates in the range of 90% to 97%. LSR values are relatively comparable to the case where lookup requests are not intercepted by malicious peers through different iterations in [GRS+14].

Figure 4.5c provides the average message overhead exerted by EM. Measurements average between 41 and 53 which is higher than MC values provided in case study one.

In Figure 3.5c, the average number of poisoned peers is provided. PR averages between 2%-7%, which shows a significant decay compared to case study one where the average ranged between 78%-90%.

As shown in Figure 4.5b, the average number of malicious peers per RT remarkably decreases as values average between 0.2%-0.6% regardless of MI. Figure 3.5e depicts the average ratio of peers per Q. Note that data collection time starts at $t = 300s$ and the EM is set to be triggered after 1800 sec. The ratio of benign peers increases from 10% to 90% as the EM is continuously triggered while malicious peers ratio decreases from 72% to 3%.

Interpreting the results

A significant increase in LSR is shown in Figure 4.5a due to the effect of EM. As malicious peers are evicted and poisoned peers are restoring correct entries towards p_v, the number of successful lookups increases. This denotes that the reliability of p_v's service provision is effectively restored when EM is activated.

As shown in Figure 4.5c, MC during the different removal procedures depicts reasonable overhead given the amount of lookups initiated and the restored reliability. As poisoned peers continuously attain correct RT entry towards p_v and malicious peers are evicted due to EM, the number of suspected peers in S decreases. Consequently, less overhead is exerted on the overlay due to EM, which is the reason MC maintains a steady average even when MI increases. Given that the number of peers in Q depends on α, MC decreases when choosing less value for α.

As depicted in Figure 3.5c, PR decreases as an effect of the notification message sent to peers during the quorum formation procedure. Besides, less peers are subject to poisoning as EM evicts malicious peers from the overlay. Accordingly, more poisoned peers are able to restore their benign status.

Moreover, EM allows poisoned peers to recover independent of the FR parameter as the large number of initiated quorums allows a large fraction of poisoned peers to exist in S, although FR determines the ratio of malicious to poisoned peers that might be suspected by the detector. The number of quorums initiated when a high detection rate is observed reaches up to 96% from the total number of initiated lookups. Once a large fraction of malicious peers are evicted from the overlay, average quorum initiated settles around 0.6%.

As illustrated in Figure 4.5b, MRT decreases as a result of evicting malicious peers at high rates as 96% of lookups initiated trigger EM. In addition, malicious peers are effectively evicted since the monitoring procedure is designed such that malicious peers are not aware of the monitoring peers or the monitoring timings, which yields no gain for malicious peers to lower FR value.

As shown in Figure 3.5e, the ratio of benign peers continuously increases in Q due to the effect of malicious peers being evicted and poisoned peers restoring their benign status. Subsequently, malicious peers existence in the quorum decreases around 0.5%. Regarding churning peers, out of $\alpha = 9$ peers that may form a quorum, the number of churning peers average around one peer per Q due to the effect of P-500 churn model. Also, as malicious peers target maximizing lookups interception, their in-going bound expects high rate

of receiving lookup requests. For this reason, removal procedures are executed in a small time-frame as time-stamps are closely assigned to quorum peers and hence, churning peers do not affect the eviction process.

3.5.6 Summary

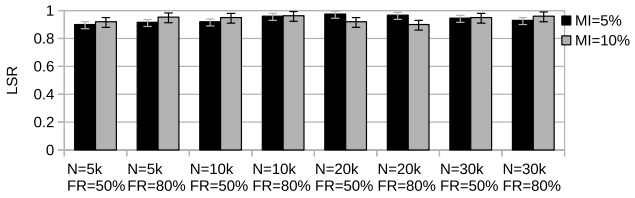
In case study one, it was shown that LAs severely impact on the overlay as LSR values drop below 1% due to malicious interception. Moreover, PR significantly increases to 90% while MRT averages between 20%-26%.

A remarkable enhancement in the overlay's performance due to EM is observed as LSR values increase to 97% while PR and MRT dropped to 2% and 0.2%, respectively. Simultaneously, MC values average around 41-53 message for different network sizes and various malicious insertions. EM stable performance on different overlay size yields its ability to be deployed in large-scale applications that host thousands-millions of users.

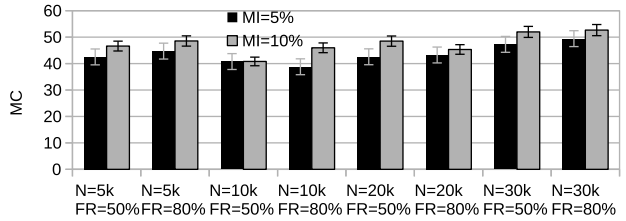
3.6 Conclusion

In this work, we propose a malicious Eviction Mechanism (EM) for P2P overlays that can efficiently perform against various LA models using decentralized quorums to monitor, decide and accordingly propagate and evict malicious peers. Our comprehensive simulation experiments shows a high malicious removal rate of up to 99% while restoring the overlay's reliability to 97%. The studies were conducted using a generic LA model that includes an established attack variety including Sybil, Eclipse attack, and more.

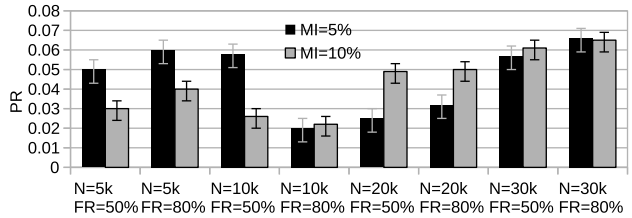
We also highlight the interest in assessing the EM performance on real P2P networks using Planet-lab. Progressively, as we show in Chapter 5, we tackled a different variant of EAs, namely OEA (where malicious peers target intercepting out-going messages) on super-P2P networks. Finally, we refer the reader to the fully detailed conclusion in Chapter 7 that connects and concludes the contributions to the respective research questions.



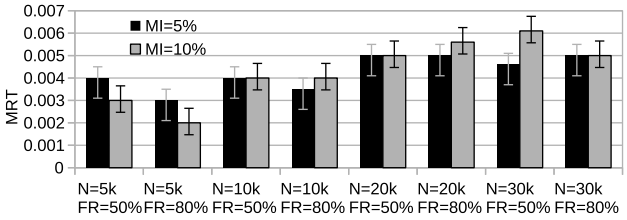
(a) Lookup success rate (LSR)



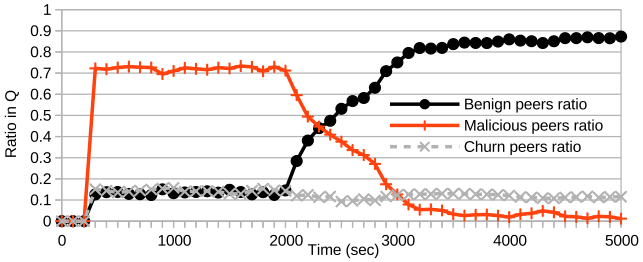
(b) Message overhead complexity (MC)



(c) Poisoned Replies per lookup (PR)



(d) Malicious ratio per RT (MRT)



(e) Peers ratio in Q

Figure 3.5: EM performance

Chapter 4

Structured Overlays: A Sanitizing Mechanism Against RTP Attacks

In this chapter, which addresses Research Question 1 and the corresponding Contribution 2, we focus on evaluating the severity of Routing Table Poisoning (RTP) attacks on structured overlays. This work was published in [IGS17]. Subsequently, we propose a sanitizing mechanism to expel malicious peers from benign peers Routing Table (RT).

4.1 Defending against RTP Attacks

In order to support scalability and low overheads in P2P networks, the design practices typically result in partitioned groups where a peer has only a partial view of the network as obtained from its neighboring peers. However, the aforementioned design practices render P2P networks susceptible to various attacks, e.g., RTP attacks, which are an inherent part of composite attacks such as Eclipse (EA), sybil, flooding and publishing attacks [KLR09; LMS+10; Dou02; GRS+14; SND+06; LYL14]. While the fault tolerance aspect ensures correct operation even for high rates of random peer failures, the disruptions inserted into peers routing tables (RT) as a form of RTP result in significant degradation of the network services. Notably, using a detailed simulation study, we demonstrate the significant RTP impact of up to 65% message loss. Moreover, we illustrate how the propagation of malicious RT information about the victim peers of RTP attacks further facilitates launching Eclipse, Sybil and other aforementioned attacks.

The existence of RTP attacks and the resulting degradation have received attention [LMG+10; CCF+12; UPS11]. A considerable variety of proposed countermeasures [LKC+12; KLK+12; LC08; CDG+02; RSV+15; SMK+01] exists, yet these techniques entail one or more of the following inefficiency drawbacks: (i) They are only applicable for a specific P2P protocol, i.e., the countermeasure mechanisms are specifically tailored according to a single P2P protocol specifications. (ii) They are effective against a single form of RTP attack. Hence,

countermeasures show no resiliency once the attack is modified. (iii) They typically require a central entity that coordinates the detection, monitoring, and decisions about malicious peers. However, in practice, the system's services are degraded as the overlay's fully distributed architecture is compromised. (iv) They often rely on cryptographic schemes, which can then constrain communication between lightweight peers to necessitate enhanced computing.

Aiming towards finding a general solution to overcome the aforementioned deficiencies, we explored a detection and sanitizing scheme in Chapter 3 as a countermeasure against a single attack variant of Localized Attacks (LAs). In this chapter, we build upon the basic notions of providing anonymous detection from our proposed mechanism in [IGS16] to develop a generalized attack handling approach applicable to multiple attack models and overlays.

4.1.1 Contributions:

In the course of our previous work, we develop an adaptable RTP attack mitigation approach that overcomes the aforementioned deficiencies. We propose a protocol-independent, fully distributed, simple and effective detection and overlay-sanitizing mechanism.

As a mean of an adaptable mitigation, we make use of a majority voting-based detection in order to detect inconsistencies in RTs. Our detection mechanism shows high accuracy with detection rates up of to 90% even for 20% malicious peers attacking. The sanitizing mechanism is triggered by initiating a quorum of peers in order to unveil the inconsistencies stemming from RTP attacks. Once the quorum investigates and accordingly declares finding malicious RT entries, the sanitizing mechanism informs other peers in order to let them reliably remove the RT information inserted by the suspected malicious peer.

Overall, our contributions span (i) demonstrating the high impact of RTP attacks on benign peers RTs and the overall network's service provision, and (ii) proposing a novel quorum based sanitizing mechanism that efficiently removes malicious peers and propagates information about their identity while providing anonymity and scalability.

This chapter is organized as follows: Section 4.2 presents the technical background along with related work. Note that the system model for structured overlays provided in Section 2.3 is used in thr course of this work. The attack model is presented in Section 4.3, followed by the description of the detection mechanism provided in Section 4.4. The proposed sanitizing mechanism is detailed in Section 4.5. The attack severity, mitigation efficiency, and detection rates are evaluated in Section 4.6, followed by providing a conclusion to this work in Section 4.7.

4.2 Related Work: Typical Attacks & Mitigation Approaches

Given the diverse set of applications that utilize the P2P functionality, a corresponding variety of attack types exists threatening the operations and reliability of P2P services. However, as routing constitutes a core P2P functionality, naturally most threats stem from deliberate attempts to compromise the peers

routing tables with malicious information. Consequently, the launching of RT attacks on P2P networks has attracted considerable research interest.

While a variety of countermeasures are proposed, most existing techniques either address a specific P2P protocol or a specific adversarial behavior arising from the malicious side of the network. To that end, we discuss (i) the existing work that addresses the impact of related attacks and the feasibility of inserting malicious peers in peers RT, (ii) the existing mitigation, detection and sanitizing techniques and their respective pros and cons, and (iii) the main aspects and challenges for the development of the generalized attack handling mechanisms. We highlight the factors that lead to providing a generalized sanitizing mechanism (as in our proposal) for malicious information removal.

4.2.1 Contemporary Approaches

We categorize each of the following presented related work according to their objective and developed techniques. For each of the presented papers, we summarize the discussion listing the pros and cons for each of them, i.e., whether such mechanisms allow for malicious peers detection, sanitizing and propagating information about malicious peers or not.

4.2.1.1 Specific mitigation techniques

Here we discuss all the relevant mitigation and detection mechanisms. This category contains mechanisms which are: (i) only effective against a certain attack, (ii) only applicable in a specific topology, (iii) dependent on the lookup approach used or (iv) only consider secure routing mechanisms as a solution.

The authors in [NW06] present a technique termed SALSA to increase lookups successful rate in the presence of malicious peers. SALSA organizes the address space into groups, thus, each peer has a limited view of the overlay. Subsequently, an anonymous forwarding scheme is used to reliably deliver lookup requests while lowering the probability of malicious peers intercepting the lookup requests. Although the proposed technique shows a successful lookup delivery rate of up to 89%, a remarkable false negative rate is noticed where malicious peers can actually bias the lookup replies. In fact, experimental studies in [MB12] show that SALSA technique can be greatly compromised when the number of malicious peers averages around 20%. Moreover, the proposed technique is only applicable in structured overlays and no sanitizing is proposed. In fact, this work is relatively comparable to our mitigation scheme proposed in [GIS15], where the mitigation approach is based on modifying the lookup forwarding protocol to prevent malicious peers from intercepting lookup requests.

Another countermeasure for RT pollution attacks was introduced in [RSV+15] that addresses P2P attacks in Smart Grids using auxiliary RT in Chord protocol [SMK+01]. However, the mitigation technique is not generalized as it is only applicable for P2P Chord based networks.

A random walk technique for structured overlays is proposed in [MB09], where peers periodically sign and certify their neighbors. Eventually, a secure route can be established for forwarding lookups. In [GRS+14], we proposed a similar random walk strategy to mitigate topology aware attacks, which yields reliable lookup delivery in trade of imposing lookup forwarding overhead on the overlay. Although these mitigation techniques effectively increase the overlay's

reliability, they do not provide detection or removal schemes and are specific to either a specific overlay topology or a specific attack variant.

In [DM09], the authors propose a labeling mechanism named SybilInfer that identifies honest and malicious peers. SybilInfer relies on a probabilistic model of social networks. Analytically, SybilInfer efficiently provides high accuracy. However, the proposed mechanism is not applicable for traditional P2P networks as the mechanism assumes that the network is aware of social connections between users, which is true for a specific subset of P2P topologies.

Usphere [KAL+15] is a countermeasure mechanism against Sybil attacks that is based on a location-independent routing protocol. Usphere relies on the trust edge created by each peer towards its 1-hop neighbors. Although the results show high resiliency against Sybil with a path stretch of O(1), the proposed mechanism has the following drawbacks: (i) relies on elliptic curve cryptography [Ber06], (ii) is only applicable to social-P2P based networks and (iii) considers neither any sanitizing mechanism nor any propagation of information about detected malicious peers.

4.2.1.2 Centralized authorities-based mitigation

A secure routing approach was introduced in [CDG+02] via encapsulating certificate authorities to peers' IDs during joining the network. Nevertheless, the proposed scheme relies on a centralized encryption authority.

In [ST15], a detection mechanism against Sybil attacks is proposed based on calculating trust values for each peer joining the overlay. However, the mechanism is effective against a single attack variant, relies on central authorities and no evaluation is provided.

4.2.1.3 Anonymity-based mitigation

In [IGS16], we proposed an anonymous detection and eviction scheme as a countermeasure to Localized Attacks where LAs refer to attacks that only target a certain set of victims. Through the detection mechanism, peers are able to make a collaborative decision along with the other peers who received the lookup request. Hence, malicious peers are removed from the benign peers RT's. Given that our approach was focusing on a specific type of attacks, the detection and the sanitizing criteria can be characterized as being (i) attack specific, and (ii) with an absence of a rapid information propagation scheme to accelerate sanitizing the overlay from malicious peers that would, in turn, increase the cost of conducting an attack. Nevertheless, using the insights developed over [IGS16], we build our generalized sanitizing mechanism utilizing the basic techniques of anonymous detection and attack sanitizing. The proposed sanitizing mechanism is evaluated against general attack scenarios, where general attack denotes common adversarial behaviors that constitute most of the well established attack forms in P2P networks.

As anonymity plays a major role in maintaining security in P2P networks and have a direct impact on the distribution of information and scalability, in [MTH+09], an anonymous low-latency networking protocol called Torsk is presented. Through the efficient relay selection and root verification schemes between peers, Torsk manages to mitigate various common P2P attacks in the Tor structured network [**loesing2010case**] while allowing the network to scale.

Similarly in [PRR09], the authors present NISAN, an anonymous approach that assures high scalability while anonymously distributing the network information. In fact, NISAN shows high resiliency to known P2P attacks. Nevertheless, the absence of detection and sanitizing schemes might put the overlay at risk if the attacker manages to insert more malicious resources in the overlay or gain newly joined peers' trust during verification. In addition, the proposed protocol is only suitable for structured overlays.

In [SND+06], the authors proposed an anonymous auditing scheme to mitigate eclipse attacks. The auditing scheme focuses on monitoring the ingoing and outgoing bounds of each peer and thus, detects peers that exceed a given threshold. Although the proposed technique allows for detecting maliciously behaving peers and locally removes those peers from the RT, no propagation or collaboration between peers to advertise such information about malicious peers exists.

In [SF12], the authors propose a partitioning scheme for large-scale overlays called Commensal cuckoo. Thus, such small groups cooperate to keep the group's decision correct despite of the launched join-leave attacks. Depending on several mechanisms such as secure routing, group authentication and bootstrapping, the proposed technique shows high resiliency even when higher fraction of malicious peers than the average state-of-the-art values exists. However, the proposed technique does not allow for propagating information about malicious peers and only addresses a single form of attack (join-leave).

4.2.1.4 Reducing complexity and overhead-based techniques

The authors in [YKG+10] discuss the overhead imposed by different mitigation and detection techniques in Distributed Hash Tables (DHTs) and how impractical these techniques are when applied to real world applications. Moreover, they present a technique to bound the message complexity when distributed quorums are required. Although the proposed technique remarkably lowers the overhead compared to already existing quorum-based techniques, the proposed technique: (i) uses a complicated cryptographic scheme and (ii) assumes that malicious peers in each quorum can maximally be $< 1/3$ of the quorum size.

A mitigation approach is proposed in [KT08]. The authors present a recursive algorithm that can reliably locate resources in the presence of malicious peers. Although the evaluation of the algorithm's performance yields very high accuracy in locating resources, the algorithm is protocol dependent. In addition, no removal or propagation of information about malicious peers from benign peers RT is provided.

4.2.1.5 Attack assessment

In [LMG+10], the authors launch an RT poisoning attack on DHTs by attacking nodes close to the victim. During the search process, malicious nodes were able to intercept and thus, manipulate the replies. Similarly, in [LKC+12; KLK+12], the authors implement an RT pollution attack in kAD, a Kademlia-based network [MM02], via allowing malicious peers to manually select keys that match the key of the victim. Consequently, malicious peers receive lookup requests directly and in turn falsely convince the lookup initiator to trust their replies.

Nevertheless, both attack mechanisms address only convergent approaches and no practical removal or mitigation techniques were proposed.

In [LC08] the authors propose an RT poisoning technique based on altering the "Hello" request messages in Kademlia-based P2P networks such as KAD. However, neither a detection nor a mitigation scheme is provided. The impact of launching RT attacks on Pastry based P2P networks is provided in [EMG14], but no further countermeasures were proposed.

Similarly, the authors in [WTC+08] evaluate the impact of several well-known attacks on the KAD network. In addition, they propose a new attack that exploits the main features of index poisoning and Sybil attacks. However, no detection or sanitizing schemes were proposed.

The authors in [NR06] point out the severity of attacking peers RT in DHT systems through proposing a DDoS attack to overload the key resources at the victim. Mainly, malicious peers manipulate benign peers to insert multiple entries in their RT with the same IP address of the victim which in turn flood the victim with messages. Similarly, in [LNR06; GK14], the authors highlight the severity of RT poisoning. However, in both works, no sanitizing mechanism for malicious entries were introduced.

Given the above discussion, we infer that: (i) RT attacks evolve in various contexts and are capable of severely degrading the network services causing significant impairments in the network functionalities, (ii) the absence of a generalized sanitizing mechanism that does not require central coordination. Accordingly, this highlights the importance of designing a generalized sanitizing mechanism that relies neither on a protocol specific parameter nor on a central coordinating entity.

4.3 Routing Table Poisoning (RTP)

In this section, we present the fundamentals of launching an RTP attack that targets inserting and propagating malicious entries in benign peers RT. The proposed attack model constitutes the basis for evaluating the proposed sanitizing mechanism, which is presented in Section 4.6.

In order to validate the effectiveness and applicability of the sanitizing mechanism in various RTP attack scenarios, we consider a sophisticated general attack model that (i) is not only applicable for a specific P2P protocol and topology, (ii) does not target a specific victim or (iii) considers various attacker capabilities and adversarial behaviors that represent severe attack scenarios.

First, we state the attacker's target. Second, the attacker's capabilities in terms of the available malicious resources and the placement criteria are described. Finally, the adversarial behaviors of inserted malicious peers that allow for poisoning peers RT are discussed.

4.3.1 RTP Attacks Types and Targets

RTP attacks are launched by allowing malicious peers to intercept lookup requests. The RTP attack's target is defined based on the intercepted lookup request's destination, i.e., malicious peers behave adversarial or not when intercepting a given lookup. RTP attacks are launched either to generally cause perturbations to the overlay (undirected RTP) or to hide the existence (directed

RTP) of a specific data or peer, referred to as directed RTP. Both attack targets are described below.

Directed attacks

In case of directed RTP, malicious peers target only a certain victim set V such that $|V| < |B|$. The selected victim set are those peers with critical or popular content. The main target of malicious peers, inserted in the overlay, is to poison entries that point to a targeted victim set [LMG+10; WTC+08; IGS16].

As the focus of this work is to assess the efficiency of the sanitizing mechanism under severe attack conditions, we do not focus on directed attacks for the following reasons: (i) a major fraction of RT entries is not poisoned. Hence, the validity of the evaluation of the sanitizing mechanism is affected as the target of the proposed sanitizing mechanism is to remove malicious peers from RTs regardless of their targeted entries to poison and (ii) directed attacks coerce specific adversarial behaviors and thus, are not suitable to assess the generality of the sanitizing mechanism. For these reasons, we do not consider directed attacks in our work.

In [IGS16], we highlight how the proposed sanitizing mechanism against directed attacks stems from the general form of the attacks proposed in this work. We study and highlight the modifications needed to launch such attack and the relative modifications in the sanitizing mechanism. Furthermore, we evaluate the impact of such attack along with the effectiveness of the proposed eviction mechanism.

Now we discuss the challenges that arise from continuously attacking all possible peers and the specific adversarial behavior executed by malicious peers.

Undirected attacks

The target of undirected RTP attacks is to poison benign peers RT where no specific victim is targeted. In this case, $B = V$.

Unlike directed RTP, a major fraction of the benign peers RT is poisoned as malicious peers do not only target specific entries to poison. As this attack target shows more severity and thus, is suitable for evaluating the performance of the sanitizing mechanism, we focus on undirected RTP as the attack's target. The target of undirected RTP attacks is to intensively pollute benign peers RT ($b \in B$) via blocking, altering or diverting lookup requests.

4.3.2 Attacker Capabilities

Now we detail the attacker's capabilities to launch an RTP attack. Capabilities refer to the amount of available malicious resources and the placement of malicious peers according to the selected undirected RTP attack.

Malicious resources

As the amount of malicious resources inserted in the overlay increases, the perturbations that can be imposed on the overlay also increase. In order to validate the performance of the sanitizing mechanism in severe attack scenarios, we assume the attacker is capable of inserting various amounts of malicious peers up

to 20% of the whole overlay size. The malicious insertions are done by inserting new peers into the network or hijacking existing peers.

Malicious placement

The placement of malicious peers mainly depends on the lookup forwarding approach specified in the P2P protocol. Hence, we start by defining the lookup approach and the matching placement criteria.

In order to validate the applicability of the sanitizing mechanism for various P2P protocols, we make use of a lookup approach that does not impose any specific criteria or route for forwarding lookup requests, i.e., to increase the attack's impact via allowing maximum interception of lookups when malicious peers are inserted. Therefore, the divergent PASS lookup approach from our previous work in [**divpass**] is usable here. The reason for selecting divergent PASS is that the peers get randomly selected within a specific address range for forwarding lookup requests.

Accordingly, the malicious peers are randomly placed within the specified range. In turn, lookup requests are equally probable to be intercepted by a benign or a malicious peer depending on the amount of inserted malicious peers.

4.3.3 RTP Adversarial Behaviors

To emphasize the impact of the attack, we propose a variety of adversarial behaviors, where malicious peers are capable of dynamically altering the actions taken according to the attacker's resources and target. We note here that since a lot of the existing research addresses the problem of join-leave attacks such as [SF12], we consider this attack behavior out of scope of this thesis.

Malicious peers attack the lookup mechanism by intercepting lookup requests and hence, replying with malicious information which affects the lookup reliability, integrity and confidentiality. Consequently, malicious information propagates to benign peers RTs causing an RTP. Once a malicious peer $p_m \in M$ successfully intercepts a request, p_m replies with an *Fake Reply (FR)* as a resolving address to the lookup request.

In an FR, p_m inserts the contact information of another colluding malicious peer which claims to hold the key of the lookup destination that p_i is requesting and falsely convinces p_i that the request was successfully resolved. As a result, $p_i \in B$ updates its RT with the newly received entry which allows malicious information to propagate through the overlay and thus, poison benign peers RT. We now define the content and the frequency of generating an FR reply.

Generating False Replies (FR)

Malicious peers are assumed to always reply with colluding malicious information to the lookup initiator. This adversarial behavior is chosen when the attacker's main target is to propagate malicious routing information regardless of the detection likelihood. This denotes that, in case of malicious peers generating false replies based on a certain probability, the detection and thus, the sanitizing of the overlay would require longer time.

Nonetheless, the perturbations effect on the overlay will be relatively less compared to the perturbations caused by malicious peers continuously lying.

We refer to our previous work in [IGS15] where the detection accuracy in case of randomly lying malicious peers is evaluated.

Note that another reason for choosing this behavior is that the attacker may target fast spreading of perturbations in the overlay. Such case can occur when the attacker owns enough resources and the attack is time dependent, i.e., the attacker's aim is to launch the attack in a specific time period or during a time triggered event in the overlay. Hence, as our focus is to evaluate the effectiveness and the applicability of the sanitizing mechanism in drastic attack scenarios, we assume that malicious peers always generate fake replies.

False Reply (FR) Content

Malicious peers control the content that should be sent in a fake reply. The possible FR content can potentially cover:

1. Replying to all intercepted lookups with a **single** malicious lookup reply. Such behavior is executed when a specific information needs to propagate through the overlay. Nevertheless, such adversarial behavior makes p_m more susceptible to detection.

2. Replying with **different** malicious replies. As malicious peers collude, p_m can reply to the lookup request with selecting one of the malicious peers that p_m is aware of. Such behavior is deployed by p_m when seeking general perturbations. In addition, replying with different malicious fake destinations further complicates the detection process.

Further details about the detection procedure of the consistent malicious replies are provided in Section 4.4.

4.4 Detection Mechanism

We now introduce the detection mechanisms used locally by each peer to suspect other peers based on the received lookup replies. In order to detect lookup inconsistencies, we propose a modified lookup mechanism in [IGS15] where peers are able to gather more than a single reply.

The lookup initiator can detect inconsistencies through comparing the set of received replies according to (i) the consent of the replying peers' location with the lookup protocol specifications, (ii) the average number of hops experienced by the lookup reply compared to the recorded average from previous lookups and (iii) the returned contact information in the lookup reply. Consequently, peers are suspected when violating these criteria. The most important feature in our detection mechanism is comparing replies, i.e., peers are also suspected when replies are not identical which are detected through a certain feature in the detector discussed through this section.

Originally for a given lookup mechanism, the lookup is terminated once the first reply that contains the requested information about a given peer p_v is received . This coerces the lookup initiator p_i to accept the lookup result without being able to validate the results since only a single reply is considered. As a result, whenever a malicious peer receives the request and replies with a fake reply, p_i accepts the reply which results in poisoning p_i's RT with a malicious

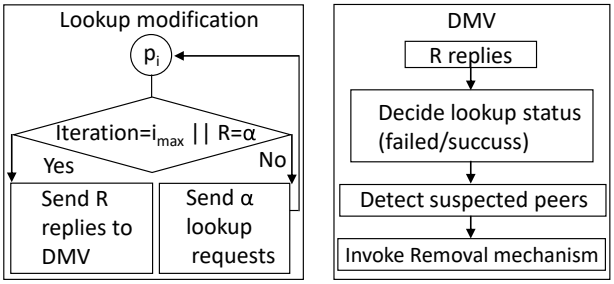


Figure 4.1: Detection procedures.

entry. The modified lookup mechanism is discussed below which provides the operations that allows p_i to gather a set of lookup replies from different peers.

4.4.1 Modified lookup Approach

We first outline the drawbacks of the contemporary lookup mechanisms, which highlights the motive of using a modified lookup mechanism. Subsequently, the operations of the modified lookup approach are presented.

The drawback of existing lookup approaches

In prior lookup implementations, p_i picks α candidate peers from its RT to start forwarding a lookup request for p_v's contact information, where α is a lookup specific parameter for the maximum number of parallel requests that can be sent. Once the lookup request is received by peer p_r, it replies with p_v's contact information if p_r has an entry for p_v in its RT. Otherwise, p_r inserts a list of potential candidates that, according to the lookup specification, have a high chance of owning p_v' contact information in their RT. Iteratively, p_i initiates α new requests from this list. Finally, the look up process terminates immediately once p_v's address is resolved or i_{max} iterations are reached.

Such approach coerces p_i to accept the single received reply. This means that p_i has no comparing base to validate the received reply, i.e., no multiple replies to enable p_i to detect inconsistencies. Hence, malicious peers can misuse this approach to falsely terminate the lookup without being detected while simultaneously resulting in a very low overhead for the lookup process.

To obviate the above mentioned drawback, we proposed a lookup modification in [IGS15]. As depicted in Figure 4.1, the major extension in the lookup modification is that the lookup process is not terminated till: (i) R replies containing p_v's contact information are received or (ii) i_{max} iterations are reached, where in this case, $R \leq \alpha$. Consequently, p_i is able to compare multiple replies from different peers and thus, detects inconsistencies to suspect certain peer(s). Note that, in this context, a suspected reply refers to a suspected peer as replies are only accepted from distinguished peers, i.e., no peer can provide more than a single reply.

For this purpose, we make use of a Dynamic Majority Voter [BJ91] (DMV) to process the R received replies. Now we discuss the technical aspects of the DMV, for more details we refer to our work in Chapter 2.

4.4.2 Dynamic Majority Voter (DMV)

Firstly, the suspected peers reported by the lookup modification are added to the set of suspicious peers S. Second, the remaining unsuspected peers are processed as inputs to the DMV. Afterwards, the DMV decides whether a valid (non-empty) majority of identical replies exists or not.

The DMV declares the lookup status as successful if such majority is found. Consequently, p_i stores the corresponding contact information. In case the lookup is declared to be unsuccessful, p_i initiates a new lookup to resolve p_v. To sum up, we provide the acceptance cases for the DMV:

1. $R \geq 3$ and a valid identical majority of the replies exists.
2. $R = 2$ and both replies are valid and identical.
3. $R = 1$ and the reply is valid.

Notably, the unmatched minority is added to the suspicious list, i.e., the remaining set of replies that did not constitute the majority. Finally, once the list of suspected peers is announced by the detector, the sanitizing mechanism is triggered to further investigate and thus, sanitize the overlay through removing malicious peers as discussed in the next section.

4.5 Sanitizing Mechanism (SM)

In this section we present the sanitizing mechanism. Prior to the operation of SM, the detector proposes a set of suspected peers according to their lookup replies as discussed in Section 4.4. Afterwards, the SM is invoked to investigate and thus, reach a decision about the suspected peers. Consequently, the SM executes a removal procedure for suspected peers identified malicious to sanitize the benign peers RT.

Unlike the detector which is operated locally by peers, SM is executed as a distributed quorum to reliably investigate and propagate information about malicious peers. Such propagation further accelerates the sanitizing rate for the overlay. Hence, the SM results in a stable and reliable P2P service provision.

We note that the quorum needs to be obtained in the decentralized P2P environment and in the presence of malicious peers. Accordingly, byzantine resilient SM procedures are developed. The sanitizing mechanism is constructed from four main procedures where each procedure is illustrated in the following subsections. For consistency, Table 4.1 provides a list of the variables and annotations used through the rest of this section.

The first procedure (Forming a quorum) defines how p_i creates: (i) a quorum, (ii) the quorum size dependabilities, and (iii) the quorum members constraints. The second procedure (Quorum investigation) describes how the quorum investigates p_s's behavior. Afterwards, the third procedure (Reaching an agreement) handles messages exchanging and reaching a coordinated decision between p_i and the quorum peers. Finally, the removal procedure (Malicious removal) is invoked to sanitize benign peers RT from peers identified to be malicious as illustrated in Figure 4.2.

Table 4.1: RTP attacks & defenses: acronyms description

Variable	Description
p_s	Suspected malicious peer
p_i	Quorum initiator
p_v	Denotes the victim peer
Q	Quorum
p_q	Peer p in Q
n	Number of peers in Q
Q_B	Set of benign peers in Q
Q_O	Set of poisoned peers in Q
Q_C	Set of churning peers in Q
Q_M	Set of malicious peers in Q
MR	Monitoring Request
RT	Routing Table

4.5.1 Forming a Quorum

As shown in Figure 4.2a, once the detector suspects certain peers, p_i starts the sanitizing mechanism by forming a quorum. In the following, we describe the criteria for choosing a quorum size along with the constraints regarding joining peers' types raised by the existence of malicious peers in the overlay. Afterwards, the formation mechanism which describes the messages exchanged while forming a quorum and the acceptance-rejection criteria for joining a quorum is described.

4.5.1.1 Quorum size and members selection

Initially, p_i selects n peers from its RT where selected peers are uniformly distributed according to their distance in p_i's RT. This selection approach guarantees peers from all possible distances participate in investigating p_s which accelerates propagating information about peers identified malicious in the complete address space.

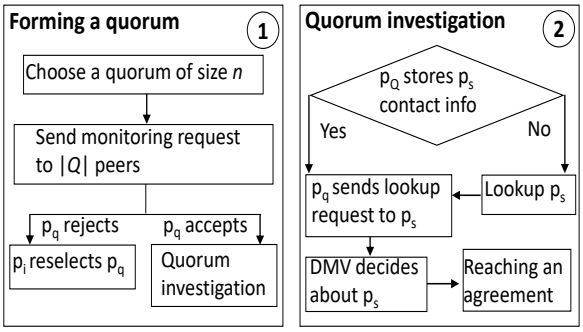
The generality of the mechanism allows to choose any distribution according to the already used lookup approach. In our experiments we use PASS, characterized by an address space divided into regions according to the common prefix in the peers' keys. Within the lookup forwarding mechanism, peers forward the lookup request to other peers located in the same address space region.

In order to propagate such information to all regions, we make use of the uniform distribution selection of quorums, so that all peers in different regions are updated with the correct information and can propagate it when requested from peers within the same region. This demonstrates that, the sanitizing mechanism is adjustable according to the basic information about the used lookup approach in the overlay.

As p_q is a member of Q, p_q belongs to one of the four sets listed in Table 4.1. Hence, n can be defined as:

$$n = |Q_{B\setminus O\cup C}| + |Q_{O\setminus C}| + |Q_C| + |Q_M| \tag{4.1}$$

where $|Q_{B\setminus O\cup C}|$ denotes benign peers that are not poisoned and do not churn. $|Q_{O\setminus C}|$ refers to the number of poisoned peers that do not churn.



(a) Forming a quorum (b) Quorum investigation

(c) Reaching an agreement (d) Malicious removal

Figure 4.2: Technical aspects of SM procedures.

In order to assure reaching a reliable decision in Q, the size of the quorum n must follow certain constraints regarding the maximum number of malicious peers that can exist in the quorum. Based on Practical Byzantine Fault Tolerance (PBFT) and Byzantine Agreement [CL99; APR+15; Fis83; HC15; CL02], the number of peers in a decentralized system that is capable of maintaining a correct state of the system is bounded by $n \geq 3f + 1$, where f is the amount of faulty peers in the system. As f peers can be malicious and another f can be poisoned or churning, this adds up to $2f$. This means that at least $f + 1$ should exhibit: (i) no malicious behavior (neither malicious nor poisoned and (ii) alive (not churning) in order to maintain a correct system state.

Similarly, the selected quorum is capable of providing reliable results in case the number of non-poisoned non-churning benign peers $|Q_{B\setminus O\cup C}|$ outnumbers the rest of other selected peers, therefore:

$$|Q_{B\setminus O\cup C}| > |Q_M| + |Q_C| + |Q_{O\setminus C}| \tag{4.2}$$

As these peers can deviate the quorum's decision according to the following possible actions:

1. $|Q_{O\setminus C}|$ peers such as $p_Q \in O \setminus C$ may provide malicious replies due to acquiring a poisoned entry from other malicious peer.

2. $|Q_M|$ where $p_Q \in M$. These peers behave maliciously via sending fake replies (see Section 4.3) to divert votes majority.

3. $|Q_C|$ peers where $p_Q \in C$ that initially accept to join the quorum. However, although these peers are neither malicious nor poisoned, they are subject to churning out and thus, do not respond to p_i.

Accordingly, the quorum size n that is capable of reliably investigating and thus, correctly reaching an agreement about suspected peers is constrained to:

$$n \geq |Q_M|+|Q_C|+|Q_{O\setminus C}|+|Q_{B\setminus O\cup C}| \tag{4.3}$$

Such constraint guarantees that the quorum's majority votes about p_s are benign, neither poisoned nor churning as depicted in Figure 4.3. In Section 4.6, we validate how such constraints hold conveniently given the existence of a remarkably high fraction of malicious peers of up to 20%. Now we describe the quorum formation mechanism in terms of messages exchanged and joining acceptance/rejection criteria.

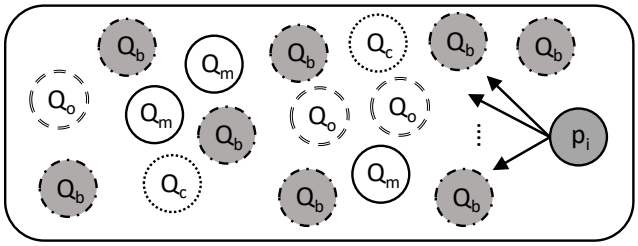


Figure 4.3: Example of types of peers in Q.

4.5.1.2 Formation mechanism

Initially, p_i sends a monitoring request message to p_q as depicted in Figure 4.2a. A monitoring request contains the contact info of each p_s. As the DMV at p_i might suspect multiple peers at the detection phase, p_i encapsulates all suspected peers in a single monitoring request which allows for the following:

1. Avoid excessive quorum formations which remarkably decreases the messages exchanged by the quorum's peers during executing the SM.

2. Increase the cost of generating fake quorums by malicious peers since the probability of detecting such adversarial behavior increases. This is due to the ability of p_q to suspect p_i after investigating the suspected peers. Hence, the SM will be invoked against p_i, more details are provided in (Malicious Removal) procedure (see Section 4.5.4).

Once p_q receives the monitoring request, it can either accept and proceed to the next step, or reply with a rejection to p_i. Rejections could be due to: (i) being involved in another quorum, (ii) loaded traffic or (iii) malicious behavior to limit quorum formations. In case p_q agrees on joining the quorum, it starts executing the quorum investigation procedure which is described below.

4.5.2 Quorum Investigation

As illustrated in Figure 4.2b, the target of this procedure is to allow each p_q to monitor p_s's behavior and thus, confirm or deny p_i's suspicion about each p_s. After receiving a monitoring request from p_i and accepting joining Q, the investigation procedure is triggered. The idea of this procedure is to check whether p_s is manipulating requests or not. Therefore, p_q monitors p_s's behavior via requesting information from p_s that p_q can validate from p_s's reply.

Nevertheless, the investigation procedure should be conducted seamlessly to obviate p_s from acting benignly once it detects being monitored. The two main aspects that define the procedure's anonymity are: (i) the type of messages p_q sends when monitoring p_s and (ii) the content requested from p_s in such a message. Both factors are detailed below.

4.5.2.1 Messages Exchange

Communication between p_q and p_s must be handled seamlessly as p_s can divert the monitoring process once it detects being monitored by p_q, which can remarkably impact on the sanitizing process.

For this reason, p_q makes use of regular lookup messages defined by the P2P protocol to avoid being detected by p_s. This means that p_q initiates lookup request to α peers, including p_s. Afterwards, p_q processes the replies to the detector to detect inconsistencies in p_s's reply.

Nonetheless, p_s can detect being monitored when it receives multiple requests requesting only p_v's contact information, i.e., assuming p_s collect statistics on how frequently it has been requested about each peer. Hence, the content of the lookup request should not reveal the victim's identity which is discussed next.

4.5.2.2 Messages content

As the RTP attack proposed in Section 4.3 targets maximizing RT perturbations, malicious peers are assumed to reply with fake replies regardless of the lookup key in the lookup request. Hence, in order not to exceed p_s's expected average of receiving lookup requesting only p_v's contact information, p_i attaches a list of peers in addition to p_v in the monitoring request. Consequently, p_q can assign any peer in the list as the lookup destination key. Using the detection mechanism discussed in Section 4.4, p_q decides whether to confirm or deny p_i's suspicion about p_s.

To sum up, the investigation procedure assures that:

1. p_i keeps the identity of the victim p_v anonymous to the quorum, which in turn prohibits malicious peers that may exist in the quorum from colluding against p_v.

2. p_s cannot predict or estimate being monitored as it receives normal lookups requesting different contact information.

3. p_q can validate its decision about p_s through the detection mechanism.

Adjusting SM: we note that our approach does not depend on a certain attack or lookup mechanism, i.e., the approach is adjustable according to the basic information such as the used lookup approach and the overlay topology.

For example, in case of the BitTorrent protocol where only one NodeID is considered, the detection is applied based on: (i) the replies' consistency about the destination ID, (ii) the average number of hops and (iii) the visited nodes compliance with the lookup forwarding criteria towards this single node. Afterwards, the detector applies the DMV to decide about the suspicious replies. Clearly, in case the fraction of malicious peers is larger than the benign ones, the detection is useless., which is detailed in [IGS15].

4.5.3 Reaching an Agreement

As depicted in Figure 4.2c, once p_q decides locally about each p_s, the decision is forwarded to the quorum initiator p_i along with a time stamp identifying the reply time of p_s. Note that p_q forwards its local decision about p_s directly to p_i instead of mutually exchanging decisions with other peers in Q. Hence, p_q have no information about the other participating peers in Q. Obscuring the identity of Q's members from each other yields:

1. decreasing the probability of malicious peers colluding against benign peers in Q.

2. preventing manipulation of the results aggregation procedure as malicious peers might intercept the aggregated results messages.

Simultaneously, p_i waits for a timeout t_{max} to receive all the monitoring replies. Otherwise, in case the waiting time exceeds t_{max}, p_i proceeds with the set of received monitoring replies. t_{max} is set according to the average time a lookup process consumes plus adding a guard time for considering the quorum formation time which is an application configurable parameter.

Afterwards, p_i inputs the set of received replies to the DMV to decide about the majority of the replies, i.e., whether the majority of the replies suspects p_s or not. Based on the DMV output, p_i reaches a decision about p_s and accordingly, inform each p_q about its decision. Now the possible decisions reached by p_i about p_s along with the consequent actions are discussed.

4.5.3.1 p_s is poisoned

p_i checks the time stamp encapsulated in each p_q monitoring reply in order to differentiate whether p_s is malicious or was poisoned. This is done through checking the time stamps sequence for each reply. In case p_s starts to consistently provide correct information about a lookup key till the last time stamp for the same key, it's considered to be poisoned. Otherwise, if p_s consistently replying or alternating with fake information, p_s is considered malicious.

4.5.3.2 p_s is malicious

Two cases lead p_i to consider p_s as malicious. First, if all the monitoring requests regardless of their time stamps report suspicious about the replies provided by p_s. Second, in case p_s's replies show alternating behavior. Alternating behavior refers to providing fake replies after providing correct ones which can be inspected from the time stamps of the received replies.

4.5.4 Malicious Removal Procedure

The agreement reached by the quorum allows p_i to determine whether to proceed with removing p_s from its RT or not. Thus, we now discuss how p_i proceeds according to the DMV decision.

4.5.4.1 Suspicion confirmed

If p_i decides that p_s is malicious, p_i executes the following **Malicious Removal (Mal-Rem)** steps as depicted in Figure 4.2d.

1. removes p_s from its RT and blocks any further contact with p_s.
2. initiates a new lookup to search for the correct contact information of p_v.
3. sends the decision to all peers in Q.

Once each p_q is informed that the quorum's decision confirms that p_s is malicious, p_q performs one of these actions depending on its local decision about p_s.

1. In case p_q's local decision also confirms the adversarial behavior of p_s, p_q executes the (Mal-Rem) steps.
2. If p_q's DMV decision about p_s' reply is benign, p_q suspects p_i and initiates a quorum against p_i.

Since p_i is aware of the conflict between both decisions, p_i expects to be monitored by a new quorum. Accordingly, in case p_i is benign, p_i consistently replies to all the lookups so that p_q detects p_i's benign status. Afterwards, p_q initiate a lookup request to p_s to investigate it's status without launching new quorums.

Without such a scheme, malicious peers will be able to invoke the sanitizing mechanism towards benign peers which can severely affect the network stability through (i) eclipsing p_v by initiating a malicious majority quorum or (ii) exhausting the network bandwidth and overload peers with exchanging messages via joining fake quorums. Such countermeasure prevents malicious peers from starting fake quorums as consequently, such malicious behavior will re-trigger the sanitizing mechanism towards these malicious peers.

Moreover, encapsulating multiple peers in monitoring requests forces malicious peers to behave benignly to avoid being suspected. Otherwise, this malicious behavior is clearly detected when p_s sends multiple fake replies at once to p_q. Thus, malicious peers refrain from initiating fake quorums.

In Section 4.6, we investigate the impact of turning poisoned entries in the quorum to benign and how propagating the correct information about the victim(s) sanitizes the overlay and increase the isolation of malicious peers.

4.5.4.2 Suspicion declined

In case p_i decides that p_s was poisoned, after processing the monitoring replies through the DMV, p_i initiates lookups to p_s requesting the contact information of the same monitoring set that was initially sent to the quorum members. Note that p_i selects distinctive time slots to initiate such set of lookups to p_s. Therefore, p_s cannot detect any abnormal behavior from p_s due to receiving excessive lookups or multiple lookups from p_i requesting p_v's contact information. Afterwards, according to p_s replies, p_i executes one of the following steps:

1. If p_s replies correctly, p_i trusts p_s and thus, inserts p_s and any reply provided by p_s in the future into its RT.

2. In case the DMV reconfirms suspicion about p_s, p_i suspects that the majority of peers in Q is malicious. To that end, p_i creates a new quorum to re-monitor and subsequently unveil p_s, after restricting peers in Q from joining the new quorum.

In Section 4.6, we evaluate the likelihood of forming a quorum with malicious majority and validate the constraints regarding forming a quorum along with evaluating the effectiveness of the proposed sanitizing mechanism.

4.6 Evaluation

This section assesses the effectiveness of SM as a countermeasure for RTP attacks launched with the set of the proposed adversarial behaviors. The target is to evaluate the severity of RTP attacks on the overlay's reliability and the imposed perturbations resulting from poisoning RT entries. Consequently, SM is evaluated in terms of reliability enhancements, imposed overhead on the network and malicious removal ratio from benign peers RT.

In order to do so, two experiments are conducted. The first experiment **(RTP attack severity)** evaluates the impact of launching RTP attacks on structured P2P overlays while focusing on how these attacks severely degrade the overlay performance. The second experiment **(Sanitizing mechanism influence)** is conducted to assess the effectiveness of SM on sanitizing overlays during launched (undirected) RTP attacks while analyzing the correlation between different SM parameters.

First we start by introducing the simulation environment, metrics and parameters used for evaluation. After that, we introduce each case study with related discussion and results interpretation. Finally, we provide a detailed summary about the results that highlight the effectiveness of SM.

4.6.1 Simulation environment

Experiments were conducted using the OMNeT++ simulator [Pon93] and OverSim [BHK07]. OverSim provides the OMNeT++ simulator with various P2P protocol implementations. For Average-Min-Max calculations, each simulation is scheduled for 10 repetitions. Moreover, for results validation, each simulation duration was scheduled to 4 hours runtime. In Table 4.2, the simulation parameters used in the experiments are provided.

4.6.2 Simulation Model

In order to validate the reliability and the scalability of the sanitizing mechanism, the experiments were conducted for different overlay sizes $N = 2500, 5000, 10000, 20000, 30000$. Moreover, high values of malicious peers $|MI|= 10\%, 15\%, 20\%$ were injected in the overlay. MI is used to measure the impairments caused in the network given a certain amount of inserted malicious resources. We now provide a description of the system workload and churn distributions used in our simulation model.

Table 4.2: RTP attack & defenses: simulation parameters

Parameter	Value
Maximum iterations i_{max}	10
Maximum replies number α	7
Key length w	128
Lookup approach	Iterative
Malicious insertion ratio `MI`	$10\%, 15\%, 20\%$
Overlay size N	$2.5k, 5k, 10k, 20k, 30k$

4.6.2.1 System Workload

In order to evaluate the experimental results for undirected RTP attacks where every peer is a potential victim, we use a "Fully Distributed Application (FDA)" workload where every peer initiates lookups requesting the contact information of randomly selected peers. Lookups are sent on average every 10 seconds with a standard deviation of 5 seconds.

4.6.2.2 Simulation Churn Models

We use a **Pareto (P-500)** churn model to simulate the churning rate of benign peers. Using the Pareto (P-500) distribution, peers acquire average life-time and dead-time of 500 seconds. Pareto distribution provides realistic experimental results for real P2P overlays scenarios [ZL06].

4.6.3 Evaluation Metrics

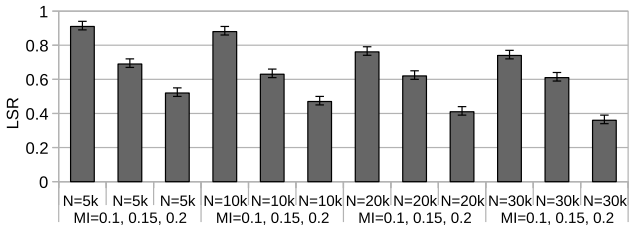
For both experiments, the following performance metrics are used.

- Lookup Success Ratio `(LSR)` measures the average ratio of successful lookups over all lookups destined to random peers. `LSR` provides insights about the reliability of the P2P overlay in both experiments.
- Message Complexity `(MC)` evaluates the average message overhead on the network per lookup. `MC` also assesses the message overhead imposed by the sanitizing mechanism.
- Malicious ratio per RT `(MRT)` provides the average ratio of malicious entries poisoned in benign peers RT as a result of the existing malicious resources in the overlay.

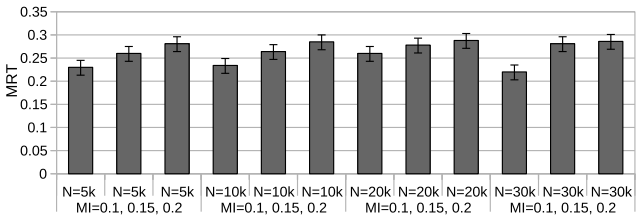
4.6.4 Experiment 1: RTP attack severity

The target of this experiment is to assess the severity of undirected RTP attacks with the proposed adversarial behaviors and to highlight the reliability degradation of the system's service provision. Results are evaluated based on `LSR`, `MRT`, `MI` and `MC`. As in undirected RTP attack, where all peers are potential victims, data is collected whenever a lookup is initiated regardless of the lookup key. We continue with describing the experimental results depicted in Figures 4.4 through 4.7, and we conclude each case study with a detailed interpretation of the results.

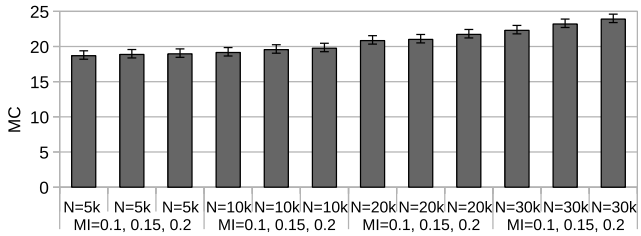
4.6.4.1 Discussion of the results



(a) Lookup success rate (LSR)



(b) Malicious ratio per RT



(c) Message Complexity (MC)

Figure 4.4: RTP attack impact.

Figure4.4a shows the LSR values across different overlay sizes. The different combinations used are: $N = 5k, 10k, 20k, 30k$ with $MI = 10\%, 15\%, 20\%$ in order to assess the correlation between inserting different malicious peers, the perturbations level in peers RT entries and the hosting overlay size. LSR remarkably decreases when MI increases as the number of malicious peers that intercept lookup requests increases. LSR values range between 36% and 91% for $N = 5000, MI = 10\%$ and $N = 30000, MI = 20\%$, respectively.

In Figure 4.4b, the average number of malicious peers inserted into benign peers RT is calculated through measuring the percentage of malicious entries to the total number of entries per RT. Note that each entry represents a distinct peer, i.e., the same peer can not exist in multiple entries. As shown, MRT increases when the percentage of malicious peers inserted into the overlay increases from 10% to 20% where the average MRT ratios range between 24% to 31% regardless of the overlay size.

The average message overhead complexity (MC) using divergent lookups is depicted in Figure 4.4c. MC measurements show comparable values across dif-

ferent overlay sizes and MI ratios, where MC ranges between 18 and 24 message per lookup.

4.6.4.2 Interpretation of the results

The significant decrease in LSR occurs due to malicious interception of the lookups. Due to the effect of RTP adversarial behavior, malicious peers reply with fake replies to p_i. Consequently, such behavior results in a non successful lookup according to the DMV decision due to the depicted majority resulted from α malicious replies.

The trend depicted in Figure 4.4a is that LSR decreases when MI increases as seen in different values of MI within the same N. Hence, at the same malicious insertions MI values and different values for N such as (combinations of $\{MI, N\}$): $\{MI = 0.15 \wedge N = \{10k, 20k, 30k\}\}$, $\{MI = 0.1 \wedge N = \{5k, 10k\}\}$ and $\{MI = 0.1 \wedge N = \{20k, 30k\}\}$ the values are closely comparable to each other. This is due to the impact of the same fraction of malicious peers on the overlay size N.

Further degradation of the success ratio occurs when poisoned peers advertise for malicious entries whenever they are queried about a lookup destination that points to a poisoned entry in their RT. In addition, as benign peers RT is polluted with malicious entries, the probability of querying malicious peers during lookup iterations increases. Hence, the corresponding ratio of benign peers queried decreases, which in turn significantly impacts the LSR.

In Figure4.4b, perturbations in benign peers RTs due to malicious peers insertion can be inferred as $MI = 10\%$ results in more than 24% poisoned malicious entries in benign peers RT. Moreover, slight increases of MRT cause significant perturbations as for $MRT = 31\%$, 64% loss rate at $N = 30$ is depicted. Although at $MRT = 22\%$ the loss rate is 26% at $N = 30$. This denotes that, with a few resources, RTP attacks are able to cause substantial impairments in the overlay.

Figure 4.4c, MC shows moderate messages overhead per lookup despite of the degraded LSR and high values of MRT. Note that the average MC overhead due to divergent PASS averages between 10-12 messages as evaluated in our previous work in [GIS15]. Nevertheless, due to the modified lookup approach, where the lookup is not terminated till α replies are returned, MC overhead increases to 18-24 messages per lookup. Although α replies are required instead of one reply, PASS shows a reasonable increase in MC due to forwarding lookup requests to a specific range in the overlay where peers are most likely to store the destination's lookup address.

Furthermore, RTP attacks do not introduce more messaging overhead on the overlay since malicious peers directly provide fake replies which minimizes the average messages exchanged per lookup. However, maintaining reasonable MC does not indicate high reliability for the overlay when RTP attack is launched. Although divergent lookups show high resiliency against localized attacks, such protocol specific approaches are inefficient against RTP attacks.

4.6.5 Experiment 2: SM Influence

In this experimental study, we evaluate the performance of the proposed SM in terms of: (i) restored reliability, (ii) malicious removal effectiveness, (iii)

imposed overhead on the overlay during the sanitizing process. Moreover, we investigate the correctness and accuracy of the quorum's decision via analyzing the ratio of selected peers according to their types (benign, poisoned, churning or malicious) during quorum formation discussed in Section 4.5.1. For consistency, same overlay sizes and evaluation metrics are used for evaluation and comparison of the results with the previous experimental study.

4.6.5.1 Discussion of the results

Figure 4.5a shows the average `LSR` when SM is on. A remarkable increase in `LSR` is noticed compared to the first experimental study, where `LSR` averaged between 36% and 91%. When SM is operating, the average `LSR` ranges from 97% to 100%.

The average `MRT` is illustrated in Figure 4.5b. `MRT` values average from 0.02 to 0.04 which is a significant decrease in the average amount of malicious entries per RT compared to the first case study where values range from 0.24 to 0.31.

Figure 4.5c depicts the average `MC` overhead induced on the overlay as a result of messages exchanged during executing the sanitizing procedures. As messages overhead due to SM varies with time according to the rate at which the mechanism is invoked, `MC` varies from 22 to 710 messages.

Although the `MC` values with SM running are remarkably high at the starting phase of the SM, `MC` decreases to show similar behavior as in experiment 1 starting at time `t=2000s` where messages average between 18 and 24. For better interpretation of the results, a time line of `MC` imposed on the overlay is provided. In addition, For comparing the variations of `MC` with the first experiment, both cases where SM is on and off are shown in the same figure. `MC` is measured at each data collection point which is scheduled every 200 seconds.

In Figure 4.6, the average `MRT` in benign peers RT is measured during SM runtime, which gives an overview about the required sanitizing time given different `MI` values.

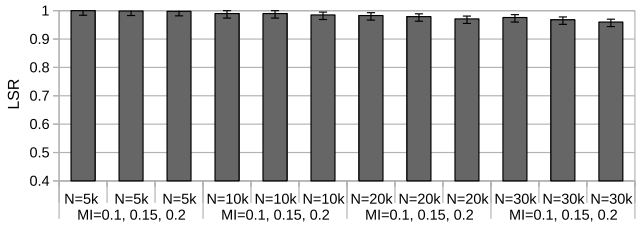
Figure 4.7 evaluates the correctness of decisions taken by the initiated quorums as discussed in Section 4.5.1 which depends mainly on the types of selected peers, i.e., the ratio of benign, malicious, poisoned and churning peers selected by p_i during the quorum formation procedure.

4.6.5.2 Interpretation of the results

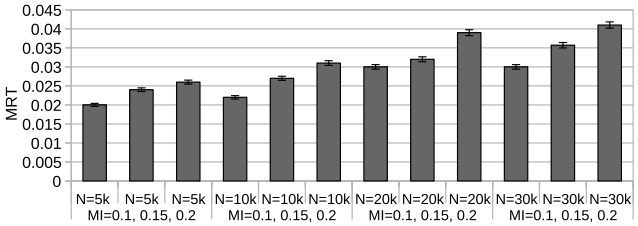
As malicious peers are sanitized from benign peers RT, the number of malicious peers contacted during different lookup iterations decreases. Hence, more correct replies are passed to the detector resulting in a remarkable increase in the ratio of successful lookups, as shown in Figure 4.5a.

Moreover, once p_Q receives a lookup request for p_v's contact information, p_Q replies with the correct information that do not contain a malicious entry. Consequently, the propagation of malicious entries decays, which in turn increases the availability of the correct contact information of peers. This results in increasing `LSR` which indicates full restoration of the reliability through SM.

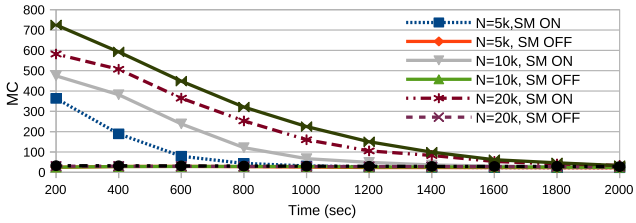
As shown in Figure 4.5b, the ratio of malicious peers that exist in benign peers remarkably decreases below 5%. This is due to the effect of removing malicious entries from benign peers RTs which consequently blocks propagating malicious entries and provide correct lookup replies to other peers.



(a) Lookup success rate (LSR)



(b) Malicious ratio per RT



(c) Message Complexity (MC)

Figure 4.5: SM performance measurements.

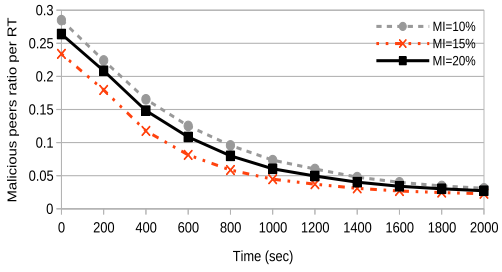


Figure 4.6: MRT decay due to SM

In addition, once p_s is announced to be malicious, all peers in Q block p_s which restrains any further contact with p_s. Hence, the attacker is not capable of restoring the ratio of malicious entries in benign peers RT using the same malicious resources. For large overlay sizes as in $N = 30000$, there is a lower probability that all malicious peers are contacted during lookup iterations. This is the reason MRT shows higher values (4%) than in smaller network sizes as for N=5000 the average MRT ratio is 2%.

Figure 4.6 provides more insights about the effectiveness of SM in decreasing MRT. Measurements are shown for $N = 10000$ and $MI = 10\%, 15\%, 20\%$. Malicious insertions are applied at the initialization phase of the network in order to evaluate the decaying rate of MRT where peers RTs are highly poisoned with malicious entries due to RTP attack. Hence, the sanitizing rate can be assessed when the attack's impact is maximized.

MRT significantly decreases below 5% even in cases when RTs are initially poisoned up to 30%. Such sanitizing rate is due to the selection of an average quorum size of $n = size(RT)/3$. Accordingly, a relatively high fraction of peers simultaneously remove p_s from their RT which accelerates the sanitizing process. Moreover, as the RTP attack is undirected, p_Q decides about p_s based on sending a lookup request destined to any random peer which allows for faster monitoring procedure as p_i assigns closer time stamps to the quorum's peers.

Note that choosing large quorum size entails a high message overhead. As seen in Figure 4.5c, the average MC generated due to quorum formations exceeds 700 messages at $N = 30000$. This is due to around 30% malicious insertions in benign peers RT which in turn increases the triggering rate of SM.

Nevertheless, MC shows a decreasing trend as MRT decreases. This denotes that the number of quorums initiated decreases as peers RTs get sanitized over time which leads MC to restore the average number of messages exchanged per lookup. In fact, a key factor of our approach is that the sanitizing mechanism do not impose permanent message overhead as the mechanism is invoked only through the detector while no permanent periodic monitoring is required.

Notably, MC depends on the quorum size. This means that selecting smaller n results in lower MC due to decreasing: (i) the number of lookups initiated to monitor p_s and (ii) the number of messages exchanged to reach an agreement. However, minimizing MC comes at the cost of a slow sanitizing rate as the number of peers joining Q decreases, which in turn decreases the number of peers that removes p_s per quorum. Subsequently, the propagation rate of peers' correct contact information is affected.

Figure 4.7 depicts peers participating in quorum formation according to their types. Such measurements are conducted for network size $N = 30000$ and $MRT = 0.28$ to evaluate the reliability of the quorum's decision in drastic RTP attack impact. As peers RT highest poisoning level is at time $t = 0$, the measured values show that the average recorded amount for benign peers $|Q_{B\setminus O\cup C}| = 0.59$ which is greater than $|Q_M|+|Q_C|+|Q_{O\setminus C}| = 0.492$.

This denotes that the majority of the replies in the quorum is benign and thus, the quorum is capable of tolerating the existence of malicious (Byzantine) replies, even at remarkably high MRT values. Hence, the quorum is able to successfully reach a reliable decision. In addition, as SM is triggered, the average number of benign peers $|Q_{B\setminus O\cup C}|$ per quorum continuously increases due to removing malicious peers from benign peers RT. Consequently, the ratio of malicious peers picked during quorum formation that affects the sanitizing mechanism decreases, which provides more reliability for Q to reach an accurate agreement about p_s.

4.6.6 Summary

Two groups of experiments were conducted in order to (i) assess the impact of launching RTP attacks on P2P networks and provide measures for the impair-

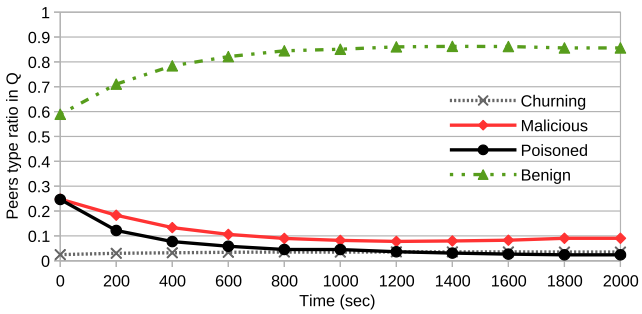


Figure 4.7: Ratio of types of peers in quorum Q.

ments imposed on the overlay and (ii) evaluate the performance of our proposed sanitizing mechanism as an approach to remove malicious peers from benign peers RT and accordingly restore the overlay's reliability.

The first experiment evaluates the impact of RTP attacks that target poisoning benign peers RT. Results show that RTP attacks severely degrade the reliability and cause significant perturbations in the overlay as peers can experience more than 64% lookup failures. Further evaluation indicates that acquiring 10% of network resources is enough for the attacker to poison more than 22% of peers RT entries.

The second experiment assesses the performance of the proposed SM. SM provides a remarkable increase in the overlay's reliability as `LSR` increases up to 100% at a reasonable, non permanent messaging overhead. Moreover, due to SM, malicious peers are removed from the overlay as malicious entries are eliminated from benign peers RT. Consequently, `MRT` drops from 28% to 3%.

4.6.6.1 SM possible challenges

One of the challenges for SM is the excessive message overhead exerted on the network when the network is drastically under attack. Given that some critical applications or lightweight peers with limited computing capability such as WSN cannot tolerate such overhead, SM might need some adjustment.

Although selecting a small quorum size n effectively decreases the message exchanged between the quorum's peers, the time delay to sanitize the overlay might be intolerant to critical applications. Hence, in order for SM to be applicable for such applications, decreasing the messages exchange between peers is a potential solution, taking into consideration the anonymity and accuracy of SM is not affected. For this, we propose to introduce a set of trusted peers in critical P2P environments that can easily manage the quorum initiation. Accordingly, the decision making procedure would result in much lower overhead.

4.7 Conclusion

RTP attacks pose a significant threat to P2P networks as reliability is severely degraded to cause service impairments. As a countermeasure, we have proposed a distributed sanitizing mechanism based on reaching a consensus once a peer is suspected over the lookup process by the DMV-based detector. The proposed

sanitizing mechanism eliminates more than 90% of the malicious entries from the peers RTs, and successfully restores the benign state of the overlay as the lookup success rate increases to almost 100%. The developed approach has been shown to be independent of the overlay structure and attack types to result in a generalized P2P attack detection and mitigation mechanism.

As a future work, we encourage assessing the performance of our sanitizing mechanism on directed RTP attacks where malicious peers manipulate replies destined to specific victims in the overlay. Malicious peers can also collude against the sanitizing mechanism once a malicious peer detects being monitored.

Besides, testing and validating the efficiency of the sanitizing mechanism on a real test-bench such as PlanetLab will be very helpful to migrate future work in that area to real deployments. Similar to the previous and the following chapters, we present in Chapter 7 a higher-level conclusion of this work while relating to the relevant research question, which is question 2 in this chapter.

Chapter 5

Unstructured Overlays: Proactive and Reactive Mechanisms

In this chapter, we address unstructured P2P overlays. The focus is on super-unstructured overlays, where a special EA attack variant, referred to as Outgoing EA (OEA), is being conducted. This work was published in [IRS18a] and counts towards Research Question 2 with the corresponding Contribution 3.

The work is presented in the following structure: Section 5.2 provides the system model along with the concepts underlying the attacker model and the conducted adversarial behaviors are defined. Section 5.3 discusses the related work and the background. The detection mechanism is fully presented in Section 5.4, while the theoretical analysis is provided in Section 5.5. The internal attack's impact, along with the detection mechanism's performance and efficiency, are evaluated in Section 5.6. Finally, we conclude our work and discuss the future work in Section 5.7.

5.1 Motivation: OEA and Super-P2P Overlays

Large-scale P2P systems for video streaming, VoIP, Massively Multiplayer Online Games (MMOG) and other popular applications host millions of users [AAM+15; GE12; WK13]. On this scale, despite their ease of maintenance, unstructured P2P systems are often inefficient as they rely on flooding or random walks to disseminate requests. Hence, the performance penalty grows rapidly with the number of nodes [LCP+05]. In contrast, structured P2P systems require considerable maintenance overhead to adapt to such scale and are vulnerable to a number of denial-of-service attacks [CCF+12; LCP+05]. Therefore, system developers increasingly rely on semi-structured *super-P2P systems* to organize service provision in P2P systems and form the foundation of large-scale applications [TLH14].

In a super-P2P system (or overlay), *super peers* disseminate messages on behalf of regular peers, which then only request or provide a service without relaying traffic. Typically, super peers exhibit an above-average performance with

regard to indicators such as bandwidth or reliability, thus improving the service quality in contrast to random peers. In addition, the hierarchical nature of a super-P2P system increases their scalability and efficiency compared to unstructured overlays as peers flood messages only between super peers. Based on a simple hierarchical network structure, super-P2P systems entail minimal topology maintenance and load balancing issues while providing inherent protection against certain denial-of-service attacks [BG03; TLH14].

However, due to their distinguished nature, super peers present tempting targets for attackers. Disabling a large fraction of super peers, which corresponds to a low fraction of all peers, effectively undermines the service provision of the overlay. The class of Outgoing Eclipse Attacks (OEAs) are particularly threatening for super-P2P systems. When launching an OEA, an attacker infiltrates the routing tables of super peers such that all or most relayed messages of these peers end up at malicious peers. These malicious peers then perform a denial-of-service attack by either dropping the requests or faking replies. Indeed, a simulation study based on real-world super-P2P systems highlighted that 0.1% of malicious peers incur a performance degradation of up to 40% [AEO12], indicating that denial-of-service (DoS) attacks might cause detrimental damage to large-scale applications that rely heavily on super-P2P systems.

Eclipse attacks and their countermeasures constitute an active area of research. However, the existing work is primarily concerned with eclipsing incoming messages in structured overlays. Therefore, most of the proposed countermeasures are not suitable in the context of super-P2P systems as they (i) rely on the routing scheme of structured overlays [FMR+09; BM07], (ii) are limited to specific adversarial behavior such as localized or topology-aware attacks [LL10], (iii) are inefficient in terms of communication or computation overhead [DKC08], or (iv) require central parties [RL03; KBC+00].

5.1.1 Contributions:

In this work, we propose an effective detection and peer eviction mechanism to mitigate OEAs in super-P2P systems. Our composite proactive and reactive mechanism mitigates the effect of routing table infiltration as well as the subsequent denial-of-service attack. Our proactive scheme aims to reduce the number of appearances of malicious peers in routing tables. For the proactive mechanism, we modify a previous auditing scheme proposed [SND+06] that enforces an upper bound on the number of connections that peers can establish. We improve the existing protocol by reducing negative impacts on honest peers that could lead to the accidental eviction of these peers in the original scheme. we provide an in-depth theoretical analysis of our modified algorithms, which provides bounds on the likelihood to recognize malicious peers within a certain time frame. Complementing our anonymous auditing scheme, our novel reactive mechanism detects denial-of-service attacks. Peers assign trust values to their neighbors and collectively blacklist peers with low trust values using a distributed consensus protocol, resulting in a permanent eviction of these peers from the system.

The evaluation of the proposed mechanism is twofold. In the theoretical evaluation, we leverage a probabilistic model to ascertain that our proactive mechanism detects malicious infiltration with high probability while erroneous removals of honest peers are rare. Combining the resulting upper bound on the

number of malicious connections with a combinatorial argument about the reactive mechanism, we provide tight bounds on the number of honest peers that an attacker can eclipse. Our extensive simulation study validates the theoretical bounds. Furthermore, our experimental results indicate that our composite detection mechanism significantly increases both lookup success ratios and throughput while only imposing overheads as low as 5%. We thus effectively mitigate a serious threat to super-P2P networks.

5.2 System Model

In this section, we present the system and adversary model. We first introduce the notation regarding the different parties in a super-unstructured overlay, followed by an overview of neighbor selection and communication protocols. Afterwards, we describe our adversary model.

5.2.1 Overlay Model

We model a P2P overlay as a directed graph $D = (P, E)$ with the set P of peers and the set $E \subset P \times P$ connections between those peers. More precisely, each peer $p_i \in P$ maintains a *routing table* RT_i containing the contact information of other peers. We write $(p_i, p_j) \in E$ if RT_i contains the contact information of peer $p_j \in P$.

Peers exchange messages to serve requests and maintain the overlay. When a peer p_i sends a request to a destination peer p_d, peers forward the message to p_d using connections $e \in E$. A *lookup mechanism* determines which connections are chosen to forward the message.

In this work, we focus on super-P2P networks; i.e., we divide the set of peers $P = S \cup R$ into two subsets of *super peers* S and *regular peers* R. We assume that super peers evolve from regular peers, where a regular peer is promoted to a super peer if it exhibits promising characteristics such as high bandwidth, CPU capabilities, reliability and storage limit.

Various super peers selection strategies have been proposed [LZL+05], though the selection strategies *per se* are not the focus here. A regular peer p_i obtains the contact information of one super peer sp_i, e.g., from a web-server or via an already known peer in the network. A super peer sp_i maintains connections to all regular peers p_i that are connected to it. We call the set of sp_i and its connected regular peers a *cluster*. In addition, sp_i maintains connections to super peers sp_j in order to serve requests from both regular and super peers.

As message delivery in super-P2P systems relies primarily on super peers, regular peers forward their *lookup requests* to super peers for dissemination. Consequently, super peers forward the lookup requests to other super peers until the destination address is resolved. Otherwise, the lookup fails.

5.2.2 Attack Model

We now describe the OEA model used for evaluating the impact of eclipsing super peers' outgoing messages. We start by stating the attacker's goal and motivation, followed by representative high-level attack strategies.

For P2P systems, we are primarily concerned with an internal attacker that infiltrates the system by inserting malicious peers. Hence, we divide the set of peers into benign and malicious peers B and M, i.e., $P = B \cup M$.

Attack Goals and Motivation We consider an internal and active adversary. The adversary aims to execute a denial-of-service attack and minimize the number of successful message deliveries. For that purpose, an attacker executes an Outgoing Eclipse Attack (OEA) on a set of victims $V \subseteq B$ with the goal of capturing all their outgoing messages. OEA is a variant of EAs: malicious peers only target outgoing messages whereas typical EAs target ingoing messages to victims. Hence, we highlight the feasibility of launching OEA as intercepting messages is a well deployed adversarial strategy. As regular peers have little to no impact on the success of message deliveries[1], we focus on malicious super peers. The attacker might either target all super peers equally, referred to as passive OEA, or focus on a specific victim set, denoted as active OEA.

In a passive OEA, the attacker has no specific set of victim super peers, therefore $V = S \cap B$. Rather, the attacker targets capturing as many messages as possible regardless of the originating peer. The reasons for launching a passive OEA include, among others, (i) a desire to degrade the overall reliability and (ii) an incapability to specifically connect to certain targeted peers.

In contrast, in an active OEA, the attacker aims to eclipse a specific subset $V \subset S \cap B$ of super peers. Complementary to the scenario of passive OEAs, the reasons for launching an active OEA include (i) a desire to prevent certain users from receiving services and (ii) the limited attacker resources to perform an attack on all peers.

Attacker's Capabilities and Strategy A decisive quantity for the strength of the attacker is the malicious fraction MI of super peers it can control, i.e., $MI = \frac{|M|}{|S|}$. We assume that malicious peers collude and are aware of all malicious peers in the network. In particular, malicious peers can observe, drop and replay received messages. Furthermore, they can send, and possibly forge, messages in the absence of a valid message authentication scheme.

Based on the above capabilities, the attacker can execute a two-fold strategy to maximize the impact of a denial-of-service attack. In the first step of the attack, malicious peers aim at receiving as many messages as possible in order to maximize their impact on the system. Being in as many routing tables as possible increases the probability to receive a message. Hence, malicious peers aim to maximize their presence in routing tables by leveraging the neighbor discovery algorithm of the overlay. Usually, they execute the algorithm at a higher frequency and accept more neighbors than intended.

Regarding the second attack strategy, when receiving a lookup request, malicious peers may either forge a reply, drop the request, or pretend that a time-out occurred and the request cannot be served. As the effect of an alleged timeout on the successful delivery is identical to the effect of dropping, we focus on the first two attack strategies.

[1]Indeed, their only option is launching a DoS attack on their super peer, which can easily be detected.

5.3 Background & Related Work

The purpose of this section is twofold. First, we summarize the existing work on EAs detection mechanisms in various P2P overlays and highlight their unsuitability for super-P2P systems. Second, we introduce Singh et al's anonymous auditing protocol for enforcing degree constraints [SND+06].

5.3.1 Attack Detection Mechanisms

There are various approaches for detecting EAs in structured P2P overlays that rely upon the existence of deterministic lookups and address-based resource allocation. For instance, Young et al. integrate a quorum into each step of the deterministic lookup [YKG+10]. It is unclear how to select the quorum members for super-P2P overlays. Moreover, an unstructured lookup involves the majority of all peers in the system, such that the additional communication overhead induced by the quorum drastically reduces the efficiency.

Furthermore, there exist malicious peer detection and eviction schemes for structured overlays based on quorums [IGS15; IGS16]. However, the proposed mechanisms consider only directed attacks, i.e., attacks launched against a specific set of victims based on their addresses in the structured overlay. These approaches are not applicable for semi-structured overlays.

In contrast, *OceanStore* [KBC+00] and *Rosebud* [RL03] offer more general protocols for malicious peer eviction in large-scale storage systems. However, both require a centralized server to handle node eviction.

Although *Commensal Cuckoo* [SF12] and *HQ* [CML+06] offer fault tolerance mechanisms, they either focus on very specific attacks or fail to evict malicious peers. Similarly, Scheideler et al. [Sch05; SS09] propose a mitigation approach against join-leave and Sybil attacks based on continuous node relocation. Though their mechanisms are efficient and provide high robustness, no reactive mechanism for detection or eviction is provided.

5.3.2 Anonymous Auditing of Node Degrees

In the following, we discuss the widely known anonymous auditing scheme introduced by Singh et al. [SND+06], that aims to limit the number of routing tables a peer exists in. Next, we highlight the key elements presented in their work that are utilized in our proactive mechanism. In the process, we identify weaknesses in the original algorithm and motivate the necessity of a new auditing scheme.

In [SND+06], the authors aim at limiting the incoming connections of peers, i.e., the number of routing tables a peer exists in. For this purpose, they propose an anonymous auditing scheme to ensure that peers adhere to an upper bound on the in-degree.

Let the *backpointer set bp* of peer u refers to the set of peers whose routing table contain u. A peer v, the *challenger*, anonymously requests the backpointer set of each peer u in its routing table. If, for peer v, a neighbor u's backpointer set exceeds the maximal allowed size or does not contain the challenger v, u does not adhere to the protocol and v removes u from its routing table. Requesting the backpointer set anonymously is the main challenge of the algorithm: peer v has to employ an *anonymizer* w to query the *responder* u so that u cannot

deduce the identity of v. We discuss the selection of anonymizer peers after elaborating on how to overcome a small fraction of malicious anonymizers.

If the responder u is malicious, a malicious anonymizer w can reveal v's identity to u so that u can include v in its reply. In contrast, if u is honest, the use of authenticated messages prevents a malicious anonymizer w from forging a reply but does not prevent w from dropping the request.

In order to account for these strategies, v executes k requests using diverse anonymizers w. If any of these requests results in a backpointer set that is too large or does not contain v, v removes u. Otherwise, v keeps u only if at least l responses contain v in the backpointer set. Under the assumption that the fraction of malicious peers is small and the peers are selected approximately uniformly, the probability to accidentally remove an honest peer or keep a malicious one is low.

Singh et al. consider several methods for selecting the anonymizer w. They rely on a secure lookup for the key $H(x_u)$ with x_u denoting the address of u. In this manner, all peers v rely on the same set of anonymizers when querying for u's backpointer set so that the anonymizer selection does not reveal information about the requesting peer. If sufficiently many peers in the set $H(x_u)$ are honest, u is unable to increase its in-degree beyond the permitted bound.

Our proactive mechanism builds upon Singh et al.'s work. However, we identify two key concerns that require careful modification of the original protocol:

1. A secure lookup for specific peers in an unstructured overlay is highly inefficient.

2. The fact that $H(x_u)$ is identical for all challengers v might lead to the expulsion of honest peers from the system and allows for individual malicious peers of arbitrary degree.

In order to clarify the second aspect, consider the scenario that more than l of the closest k nodes to $H(x_u)$ can be malicious with a certain probability. If u is malicious, more than l malicious anonymizers imply that u can convince all challengers that it is honest. Thus, u could have an arbitrary high in-degree. On the other hand, if u is honest, more than $k - l$ malicious anonymizers imply that all challengers will drop their connection to u, ultimately exiling the honest peer u. In contrast, if each challenger v uses a distinct set of anonymizers for the responder u, only a small fraction of challengers with malicious anonymizers removes u. Hence, u can still participate in the system.

In summary, the work by Singh et al. provides some key ideas on how to monitor the number of connections a node can establish. Nevertheless, the algorithm in its current is unsuitable for our scenario. As a consequence, we propose a novel two-fold mechanism that specifically overcomes the aforementioned issues. Our mechanism provides (a) a method for selecting dynamic and distinct anonymizer sets and (b) a distributed consensus protocol that expels malicious peers, which is an aspect that is not considered in Singh et al. 's work.

5.4 Detection Mechanisms

In this section, we present our proposed combined proactive and reactive mechanisms for detecting malicious behavior. The goal of the proactive mechanism

is to prevent malicious peers from gaining a disproportional influence. Complementary, the reactive mechanism detects and expels malicious peers. Table 6.1 enumerates variables used throughout this relevant work in the thesis.

5.4.1 Proactive Detection

An attacker usually attempts to maximize its influence by infiltrating as many routing tables as possible. In order to mitigate the impact of the attacker, we aim to reduce the fraction of routing tables malicious peers can infiltrate.

Our mechanism keeps the key ideas of the anonymous auditing scheme by Singh et al. presented in Section 5.3.2 but i) replaces the secure deterministic lookup with a gossiping protocol, ii) bases its decision upon a sliding window of replies from individually chosen anonymizers, and (iii) does not require a structured overlay for selecting anonymizers. We now detail these modifications.

First, we integrate a gossiping protocol in our system, such as Cyclon [VGV05]. Here, super peers periodically spread their contact information through the network[2]. In this manner, all peers eventually obtain the contact information of all other peers. Note that the overlay consisting of super peers usually is of a smaller size and higher stability. Hence, spreading the contact information of all peers is indeed feasible.

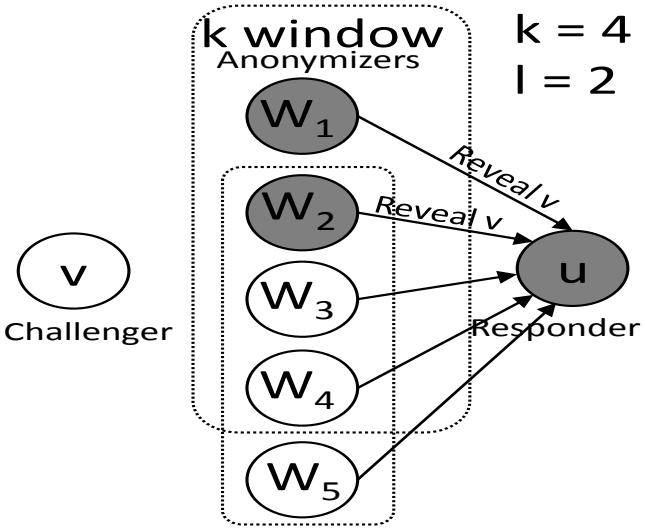
Second, we propose a new protocol for selecting diverse anonymizers using the information disseminated by the above gossiping protocol. As each challenger v has the contact information of all or most other peers, v periodically selects an anonymizer uniformly at random from all known peers. v then bases its decision on whether to remove a neighbor u from its routing table upon the last k queries; i.e., the decision is based on a sliding window of the most recent queries. In other words, v removes u from its routing table if:

- a received backpointer set is too large,
- a received backpointer set does not contain v, or
- v received fewer than l valid replies from the last k queries for u's backpointer set.

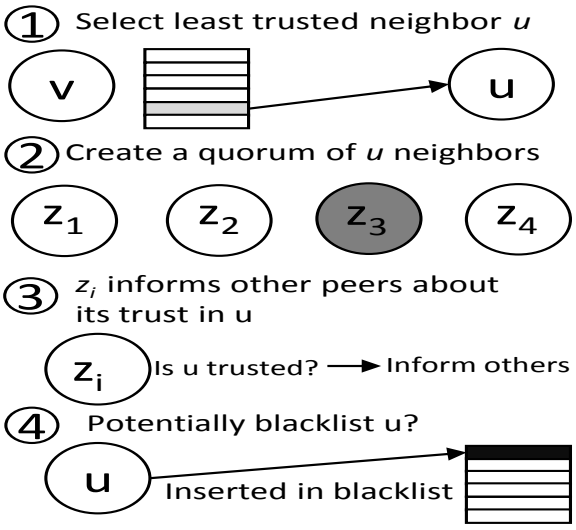
Otherwise, v keeps u for the time being but continues to periodically requests its backpointer set, see Figure 5.1a.

Malicious peers are mostly unable to bias the selection of anonymizers for such a gossiping protocol. Unless the sub-graph induced by the benign peers B is disconnected, dropping the contact information of honest peers is of little consequence as the blocked information reaches the peer via a different path. Similarly, spreading their own information at a higher frequency, malicious peers might bias newly joined peers initially. However, peers only store each peer's contact information once. As soon as a peer has received the contact information of all benign peers, the bias disappears. Thus, the peer selection is indeed close to uniform and the probability to select a malicious anonymizer peer is approximately equal to the fraction MI of malicious peers.

[2]For brevity, we write 'peers' for the rest of the section rather than 'super peers'. However, only super peers participate in the proposed mechanisms, as malicious regular peers do not have significant impact on the system.



(a) Proactive mechanism



(b) Reactive mechanism

Figure 5.1: Proactive and reactive detection mechanisms

Figure 5.1a Our proactive mechanism prevents malicious peers from occupying too many routing tables (malicious peers are marked with dark grey). Peers v anonymously request the backpointer set of their neighbors and remove those that i) reply with too large sets, ii) reply with sets that do not contain v, or iii) fail to reply too often (here: 3 out of the last 4 times).

Figure 5.1b Our reactive mechanism aims to expel malicious peers executing a denial-of-service attack. Peers construct a quorum on the trustworthiness of a peer exhibiting a bad performance.

We now explain how our modification prevent the unintentional exiling of honest peers. The continuous changing of anonymizer nodes implies that eventually one set of anonymizer nodes is dominated by malicious peers. As a consequence, an honest peer v will remove an honest peer u from its routing table as malicious peers create the impression that peer u is unwilling to respond and, hence, is likely malicious. However, v does not expel u from the system but only from its own routing table, so the removal merely requires v to select a new neighbor. Such additional edge churn is acceptable if its frequency and the associated overhead is low.

Thus, as detailed in [SND+06], the probability that more than l of k randomly chosen anonymizer peers are malicious corresponds to the probability that a binomially distributed random variable X with parameters k and f attains values of at least $l+1$, i.e.,

$$P(X \geq l) \sum_{i=l+1}^{k} \binom{k}{i} f^i (1-f)^{k-i} \tag{5.1}$$

For values of $f \leq 0.2$ and $l = \lceil k/2 \rceil$, the probability to accidentally accept a malicious peer is very low, as we show in Section 5.6.

5.4.2 Reactive Detection

Our reactive mechanism expels malicious peers based on a two-step process. First, a super peer v keeps track of the reliability of its neighbors. If a neighbor u does not perform adequately, v marks u as suspicious and initiates a quorum in order to decide upon u's continued presence in the system. The quorum consists of all peers with outgoing connections to u, i.e., peers in u's backpointer set, as the misbehavior of u primary affects these peers as they forward requests to u. Note that v uses the most recent backpointer set it received while executing the proactive mechanism from Section 5.4.1.

Second, the quorum then collectively decides to either allow the peer to remain in the system or not. This means that a malicious peer thus might remain in the system if either more than $1/3$ of its in-coming connections are from malicious peers or it only selectively misbehaves in order to fool the majority of peers into believing that it is benign. Next, we specify how i) the peer v decides to initiate a quorum and ii) to reach an agreement on the expulsion of a peer.

In order to determine a neighbor's reliability, v needs to be capable of determining if a request forwarded to the neighbor u was adequately processed by u. Usually, v cannot detect the misbehavior of u with absolute certainty but can determine if the requested service has been provided, e.g., if the correct file was retrieved. However, the lack of a positive response does not necessarily imply a misbehavior of u. Rather, a failed request can indicate a misbehavior by any other peer contacted via u to provide the desired service. Furthermore, an inability to provide the desired service, e.g., requesting a non-existent file, results in failure without the presence of misbehavior. Accordingly, individual failures to provide the desired services should not directly result in marking the corresponding peer as suspicious. Rather, we propose an iterative adjustment of the neighbors trust-values [ZSD12].

Peers assign their neighbors trust-values $tv_v(u)$, which are increased by δ_1 upon the successful completion of a request forwarded to the neighbor. On the

other hand, if such a request fails, the trust value is decreased by δ_2. Note that peers periodically check their contacts' liveliness to differentiate between offline and non-cooperative peers. Periodically, peers v select their neighbor with the lowest trust value that has not been subject to a quorum for at least time τ. v initiates a quorum to decide if the peer should be expelled.

After v has decided to establish a quorum to decide upon removing a neighbor u, v retrieves the contact information of the peers in u's backpointer set from the most recent backpointer set received during the execution of the proactive mechanism from Section 5.4.1. As depicted in Figure 5.1b, each member w of this quorum ranks the trust values of their super peer neighbors $SN(w)$ and considers $L_{w,r}$, the set of the $\lfloor rSN(w) \rfloor$ neighbors with the lowest trust value for $r \in [0, 1]$. If $u \in L_{w,r}$, then w agrees with the expulsion of u, otherwise not. Finally, w sends its decision to all other peers participating in the quorum. After a time-out, each quorum member considers the received votes. If more than half of them declare that u is malicious, all quorum members remove u from their routing table. In this manner, all honest peers in the backpointer set remove u from their routing tables.

If u is indeed expelled, the above protocol drastically reduces u's connectivity. Ideally, u does not have any remaining neighbors and cannot join again. However, malicious peers can remain connected to u and continue to advertise u as a potential neighbor to honest peers. Furthermore, the returned backpointer set might only be a subset of u's actual set.

Due to the in-degree bound, the number of additional honest neighbors should be small and restricted to peers that recently added u to their routing tables, so that those peers did not yet receive a backpointer without their entry. We thus suggest keeping a blacklist of expelled peers. In order to ensure that all peers in the list have indeed been removed, peers blacklisting others have to provide i) the backpointer set used for the quorum signed by the expelled peer u, and ii) the signed responses of all members regarding the expulsion of u. Peers can spread new entries to the blacklist via gossiping, so that eventually all honest peers remove blacklisted peers. For consistency, newly joining peers or peers changing status (offline to online) receive the latest blacklist prior to adding new contacts.

However, malicious peers can abuse the quorum mechanism to discredit honest peers. If the majority of the backpointers of an honest peer u points to malicious peers, these malicious peers can collectively decide to expel u. The remaining honest peers in the quorum remove u from their routing tables and effectively expel u from the network. Because the blacklisting algorithm requires the signatures of all quorum members, blacklisting is only possible if the peer was indeed expelled. So, the blacklisting does not allow for additional abuse. However, we show that the probability of malicious peers dominating the quorum of an honest peer is low in the following section.

5.5 Analysis

In this section, we first derive the probability of erroneously removing honest peers and correctly removing malicious peers whose degree exceeds the upper bound θ from routing tables. Afterwards, we show that our collective agreement protocol further reduces the number of edges an attacker can use to launch an

attack. Based upon this bound on the number of edges, we present an upper bound on the damage that a denial-of-service attack can cause.

5.5.1 Effectiveness of the Degree Bound

To assess the effectiveness of our approach for enforcing a degree bound (cf. Section 5.4.1), we have to consider 1) the probability of forcing malicious peers to adhere to the bound on the degree, and 2) the probability of accidentally removing peers from routing tables.

As the set of anonymizer peers changes over time and malicious peers are largely unable to bias or predict the selection, any peer v with a malicious neighbor eventually obtains a set of anonymizer peers with less than l malicious peers. In contrast, v also eventually obtains a set of more than $k - l$ malicious peers to check upon an honest neighbor u, thus erroneously removing u from its routing table. The parameters k and l should be chosen such that the duration of connections between honest peers is long, while malicious peers are removed nearly immediately. Hence, the rare removals of honest peers should barely affect the performance and allow the removed peers to obtain new neighbors easily. On the other hand, the frequent removals of malicious peers from routing tables should force them to maintain the prescribed limit θ. In the following, we evaluate the frequencies of removals for both honest and malicious peers.

We model the response behavior as a Markov chain. Each state of the Markov chain $(X_i)_{i\in\mathbb{N}_0}$ corresponds to a set of anonymizer nodes. The absorbing states of the Markov chain correspond to the removal of a peer u from v's routing table. Then, we can derive the probability that a peer v removes its neighbor u as the probability that the Markov chain has reached an absorbing state.

Formally, the state space $S = \{0, 1\}^k$ describes the distribution of malicious peers within the last k anonymizer nodes. In other words, let a vector $s = (s_1, \ldots, s_k) \in S$ represents k anonymizer nodes with $s_i = 1$ indicating that the $k - i + 1$-th recent anonymizer node is malicious. As M is the set of malicious peers, with $MI = \frac{|M|}{|S|}$ denoting the fraction of malicious super peers, we have $P(s_i = 1) = MI$ for all i and consequently $P(s_i = 0) = 1 - MI$. The initial probability of a state s is the probability to obtain a certain list of malicious and honest anonymizer nodes from the backpointer set:

$$P(X_0 = s) = \prod_{i=1}^{k} MI^{s_i}(1 - MI)^{1-s_i} \tag{5.2}$$

The transition between states is governed by two rules. The set of absorbing states A_X, i.e., states s such that the transition probability $P(X_1 = t|X_0 = s) = 0$ for all states $t \neq s$, corresponds to all states s with $\sum_{i=1}^{k} s_i \geq k-l+1$. In other words, absorbing states correspond to more than $k - l$ malicious anonymizer nodes, resulting in the removal of u from v's routing table. Transitions from a non-absorbing state to any state correspond to removing the first entry in the vector, i.e., the k-th recent anonymizer node, shifting all other entries, and adding a new entry. The new entry is either 1 or 0 with probability MI or

$1 - MI$, respectively. Hence, for all non-absorbing states $s \in S$, we have

$$P(X_1 = t | X_0 = s) = \begin{cases} MI, & t_k = 1, t_i = s_{i+1}, i = 1..k-1 \\ 1 - MI, & t_k = 0, t_i = s_{i+1}, i = 1..k-1 \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

After enumerating all elements in S, we can construct an initialization vector I and a transition matrix T based on Eqs. 5.2 and 5.3. The probability distribution of the random variable X_i corresponds to $T^i I$. We obtain the probability of reaching an absorbing state at step i as $p_i = \sum_{s \in A} P(X_i = s)$. Expressed in terms of our original question, an honest peer u remains in the routing table after $k + i + 1$ anonymizer nodes have been contacted with probability $1 - p_i$. We derive the probability for various values of k and l in Section 5.6.

Malicious peers As above, we derive the probability of removing a malicious peer u after i steps based on a Markov chain model. However, if v requests the backpointer set of a malicious peer u via an anonymizer node w, u can apply diverse strategies. Thus, the probability to remove a peer u depends on u's strategy, such that the derivation is slightly more complex.

The reaction of a malicious peer u receiving a backpointer set request depends partly on the anonymizer node w. If w is malicious, u can construct a backpointer set containing the requesting peer v and thus, provides a correct response. If w is honest, u does not know which peer initiated the request. Thus, if u's degree exceeds the bound θ, u might be unable to construct a correct response. Hence, one strategy of u is not to reply. Alternatively, u can reply with a subset of θ of its backpointers, including each backpointer z with a probability $q(z)$.

For instance, u might return the same subset W for every request, i.e., $q(z) = 1$ for $z \in W$ and $q(z) = 0$ for $z \notin W$. As a consequence, if the requesting peer $V \in W$, v is unable to detect that u does not satisfy the bound on the degree. However, if $v \notin W$, v can detect that u is malicious once it receives one response via an honest anonymizer node. Thus, peers not contained in W remove u from their routing tables instantly. Alternatively, u could select potentially different sets of randomly chosen backpointers for each request. However, as long as u maintains more than θ incoming links, the peers eventually receive a backpointer set that does not contain them and drop u. Again, the set of backpointers of u should soon be reduced to size θ. In contrast, if u does not respond, the requesting peer v does not immediately remove u but considers the responses of the last k requests. If v receives l responses in total ($l < k$), it removes u.

In the following, we assume that u responds with probability p. If u responds, it includes the backpointer z with probability $q_t(z)$. Note that the probability $q_t(z)$ to return z is likely to change over time, as the backpointer set of u is dynamic.

We now describe how to integrate the different strategies of u into our Markov chain $(Y_i)_{i \in \mathbb{N}}$. Again, our state space $S_Y = \{0, 1, 2, 3\}^k$ characterizes the responses to the last k requests. However, we now consider four different types of responses R_t, namely 0: malicious anonymizer node w ($P(R_t = 0) = MI$), 1: honest w, u does not respond ($P(R_t = 1) = (1 - MI)(1 - p)$), 2: honest

w, u's response includes v ($P(R_t = 2) = (1 - MI)pq_t(v)$), 3: honest w, u's response does not include v ($P(R_t = 3) = (1 - MI)p(1 - q_t(v))$). Hence, we can approximate the initial distribution by

$$P(Y_0 = s) = \prod_{i=1}^{k} P(R_i = s_i) \tag{5.4}$$

with R_i indicating the i-th query[3]. The set of absorbing states A_Y consists of all states for which at least one element is 3, i.e., the response does not contain v, or more than $k - l$ elements are 1, i.e., u did not respond. For non-absorbing states, the transition probabilities are obtained analogously to Eq. 5.3, removing the first element of the vector indicating the state and adding a last element corresponding to any of the four possible response types with the respective probability.

Given the initial and transition probabilities, we derive the probability that a malicious peer u remains in the routing table after $k + i$ anonymizer peers. We present results for various values of k and l in Section 5.6.

5.5.2 Effectiveness of the Quorum

In Section 5.4.2, we proposed a method to collectively expel malicious peers from the system rather than only removing them from a set of routing tables. Here, we analyze to which extent malicious peers can counteract this method by i) ensuring that quorums against malicious peers fail, and ii) discrediting honest peers to expel them from the system. Based on the results, we show that passive eclipse attacks are unlikely to have any impact. However, active eclipse attacks might result in eclipsing a small set of peers but only after remaining in the system for an extended time period.

Throughout this section, D denotes the average routing table size, which is equal to the average in-degree. In a well-performing system, D should be close to θ for maximal connectivity and performance.

Honest peers We start by considering the less complex case of discrediting honest peers. Let E denote the event that malicious peers dominate the backpointer set of size $|bp|$ of an honest peer and hence, can expel the peer from the system. Intuitively. The malicious peers have to control $\lfloor |bp|/2 \rfloor + 1$ of the pointers. As pointed out in Section 5.5.1, malicious peers maintaining more than θ in- or outgoing connections are likely to be detected soon. Thus, the routing table size of a malicious peer is essentially bounded by θ, disregarding short-time connections that are quickly dissolved when the lack of adherence to the degree bound is revealed. The total number of routing table entries of all malicious peers is bounded by $\theta|M|$, while the number of edges in the system is $D|B \cup M|$. Hence, the fraction of edges to malicious peers is at most $\theta|M|/(D|B \cup M|) = \frac{\theta}{D}MI$.

Now, we consider how the fraction of malicious edges corresponds to the probability of discrediting an honest peer. If malicious peers establish random

[3]We here assume that routing table entries are independent, which is not strictly true, as each peer can appear at most once in another peer's routing table. However, the approximation error is negligible for a large network.

connections rather than targeting individual peers, each incoming connection of an honest peer v is to a malicious peer with probability $\frac{\theta}{D}MI$. Thus, the number of connections to malicious peers in an honest peer u's bp (approximately under the assumption $\theta << n$, where n is the network size) is binomially distributed with parameters $|bp|$ and $\frac{\theta}{D}MI$, such that

$$P(E) = \sum_{i=\lfloor |bp|/2 \rfloor + 1}^{|bp|} \binom{|bp|}{i} \left(\frac{\theta}{D}MI\right)^{i} \left(1 - \frac{\theta}{D}MI\right)^{|bp|-i} \tag{5.5}$$

In contrast, if malicious peers target a set V of peers, they should eventually control a considerable fraction of incoming connections of these peers. Assuming that each targeted peer has an average in-degree of D, malicious peers need to control at least $0.5D|V|$ of the edges to peers in V to discredit them. Thus, they can execute the attack on sets of size up to $|V| = \frac{2\theta}{D}|M|$. However, super peers are usually very stable, such that the connections between super peers are likely to change slowly and thus, establishing targeted connections should take considerable time. In particular, targeted peers might leave before the attack is successful. We evaluate the duration of targeted attacks in Section 5.6.

Malicious peers In general, malicious peers can either prevent the quorum or manipulate it such that the malicious peer is not removed. However, in order to prevent quorums, malicious peers have to provide better service than some honest peers. If malicious peers provide acceptable service, their removal does not improve the overall quality of service since their attack cannot really be seen as a DoS attack. Thus, improving the (malicious) peer's performance such that quorums with regard to malicious peers are not initiated is generally not a viable strategy for an attacker aiming to degrade the service.

In order to prevent a quorum from expelling a malicious peer u, at least half of the peers in the quorum have to vote for u to stay. The peers voting for u can be either malicious or honest peers that actually received service from u. Thus, each malicious peer can show malicious behavior towards at most $1/2\theta$ peers. There are a total of $|B \cup M|D$ edges in the network, where at most $1/2\theta|M|$ edges can be used for malicious behavior. As a consequence, the fraction of edges along which peers execute malicious behavior is bound by $\frac{\theta}{2D}MI$. We call a malicious peer actually executing malicious behavior towards a peer v a *maliciously behaving neighbor*.

Now, we consider the likelihood that maliciously behaving neighbors eclipse a peer by controlling x or more of its rt routing table entries. If malicious peers do not target individual peers but accept random connections, the probability for any entry in the routing table to be a maliciously behaving neighbor is $\frac{\theta}{2D}MI$. Hence, the probability distribution X of the number of maliciously behaving neighbors is approximately binomial with parameters rt and $\frac{\theta}{2D}MI$, such that

$$P(X = x) = \binom{rt}{x} \left(\frac{\theta}{2D}MI\right)^{x} \left(1 - \frac{\theta}{2D}MI\right)^{rt-x} \tag{5.6}$$

If malicious peers target a set of peers $|V|$, they might again eventually control all $|V|D$ outgoing edges of these peers, as long as $|V|D < \theta/2|M|$. The maximal size of V is hence $\frac{\theta}{2D}|M|$. Still, manipulating an honest peer to accept new malicious peers might take a considerable amount of time, which we further assess in Section 5.6.

5.6 Evaluation

We now show the overall effectiveness of our detection mechanism in two case studies. First, we demonstrate the importance of an effective detection mechanism by quantifying the impact of OEAs on super-P2P systems. Second, we show how different parameters affect the effectiveness of our detection mechanism and its communication overhead.

5.6.1 Simulation Framework

We leverage the widely used discrete event based OMNeT++ simulator [Pon93] with the OverSim framework [BHK07]. In order to realize communication between super peers, we make use of OverSim's GIA module, short for Gianduia, for unstructured overlays [CRB+03]. GIA communication is restricted to super peers: regular peers attach themselves to one random super peer and restrict their communication to this peer only. The super peer assignment leverages existing techniques [MHC06].

Upon promotion to super peers, these peers receive a list of existing super peers, e.g., via gossiping and choose potential neighbors from this list. We assume all connections to be bidirectional, i.e., a node v has u in its routing table if and only if u has v in its routing table.

As discussed in Section 5.2, both super and regular peers initiate lookup requests to retrieve content stored in the system. Once regular peers contact their super peer, super peers use flooding to discover the requested content.

The detection mechanism is implemented based on the description in Section 5.4. For the proactive mechanism, we consider two attack strategies. If a malicious responder u with more than θ neighbors encounters an honest anonymizer, u either replies with a random (but constant over one simulation run) backpointer set containing θ of its actual backpointers (denoted ***guess-reply***) or refuses to reply (denoted ***no-reply***). We assess the impact of these adversary behaviors below.

For the reactive mechanism, we consider two denial-of-service attacks. Attackers can either drop received lookups (denoted ***drop***) or return a fake reply (denoted ***FR***). Faking replies are usually more effective as only one reply is forwarded to the originator of the request. Hence, super peers may forward a fake reply despite finding a route to the correct target. However, if the content is authenticated, peers detect fake replies and discard them. Thus, it is also essential to consider the impact of *drop reply*.

5.6.2 Simulation Set-up and Metrics

In our simulation study, we consider overlays of 3,000, 6,000, and 10,000 peers in total to validate the performance of our approach when the overlay size scales. We set the ratio of super peers to 10%, which is common for such overlays [BG03; BS06]. The maximal routing table size of honest peers corresponds to 25% of the total number of super peers and the maximal depth for flooding is 15. The fraction of malicious peers MI varies between 10% and 20%.

Each peer initiates a lookup every 4 seconds on average. Peers choose the target of the lookup randomly. However, for active OEAs (a targeted specific victim set), we limit the computed statistics to lookups addressed to peers

within this set. Our simulation time is $1000s$ with 10 runs of each parameter combination. Statistics collection is every $15s$. The proactive detection starts at $t = 100s$ and is invoked every $10s$, and the reactive mechanism starts at $t = 500s$ to allow peers to acquire trust values and is invoked every $70s$.

In our second case study, we evaluate various parameter settings for both the proactive and the reactive mechanisms. For the proactive mechanism, we consider window sizes $k \in \{6, 12, 16\}$ and required correct replies $l \in \{3, 4, 6, 8\}$. For the reactive mechanism, we choose the interval $\tau = 100s, 150s$ between subsequent quorum initiations to the same peer and the factor r to determine the list of the lowest trust value peers $r = \{0.07, 0.1, 0.15, 0.2\}$. Initially, peers assign each of their neighbor a trust value of 0 and the trust increment δ_1 and decrement δ_2 are both 1.

In this manner, we cover a wide range of overall parameter combinations. In our discussion, we focus on the most prominent results of the parameter study. We focus on the following common metrics for characterizing the performance of lookup and detection mechanism:

Lookup Success Ratio (LSR)

the fraction of successfully answered requests, averaged over all peers and lookups for passive active OEAs and all lookups originating from victim peers.

Detection Overhead (DO)

the ratio of the number of messages exerted on the overlay by the detection mechanism and the total number of messages sent or processed by a super peer.

Malicious Ratio per RT (MRT)

the ratio of malicious peers in an honest peer's routing tables, averaged over all honest peers per collection interval.

Expulsion Ratio per RT (ERT)

the ratio of correctly expelled malicious peers and the overall number of malicious peers in an honest peer's routing table, averaged over all honest peers.

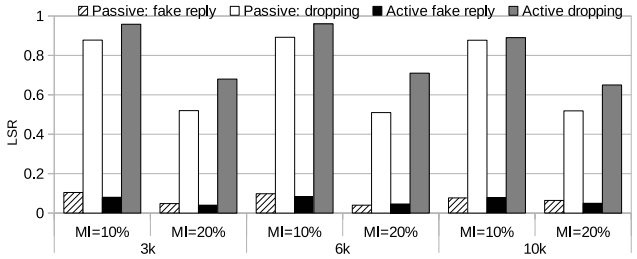
Honest Keeping Ratio per RT (HRT)

the ratio of successfully keeping honest peers when queried and the overall number of proactive initiations to honest peers.

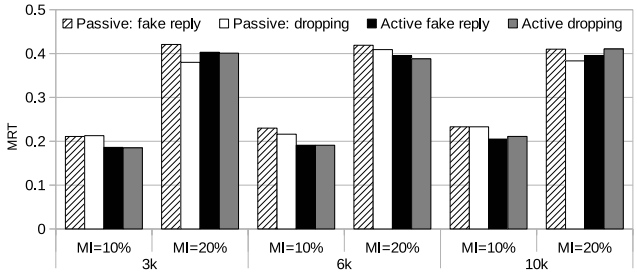
The first two of the above metrics are the main performance indicators of a well-functioning system, so our primary goal is to maximize the lookup success at acceptable detection overhead. In contrast, the latter three metrics allow us to evaluate the effectiveness of our detection mechanism.

5.6.3 Case Study 1: OEA impact

The main goal of this study is to highlight the severity of OEAs on super-P2P overlays based on the aforementioned metrics and parameters.



(a) Lookup success rate (`LSR`)



(b) Malicious ratio per RT (`MRT`)

Figure 5.2: OEA impact on super-unstructured networks

We start by considering a super-P2P system without any attacks. Without attacks, we achieve an LSR of at least 99%, more precisely 99.8% for 3,000 peers, 99.2% for 6,000 peers, and 99.0% for 10,000 peers. The slight chance of failure is due to the limited depth of the flooding, i.e., the rare cases where all peers storing a specific content are out-of-reach.

In contrast, attacks result in a low LSR, as illustrated in Figure 5.2a. In particular, FR results in a very low LSR with an average success ratio between 8%-11% for $MI = 10\%$ and 4%-7% for $MI = 20\%$. The low LSR is due to the instant termination of the lookup upon receiving a fake reply. The dropping behavior shows a higher average LSR of 80%-96% and 51%-89% for $MI = 10\%$ and $MI = 20\%$, respectively.

The reason for this higher LSR average is that, unlike in FR, the dropping behavior does not instantly results in a failed lookup. Hence, depending on the flooding value used, honest peers still may receive and accordingly respond with the destination's contact information, which results in a successful lookup. Nevertheless, the reduction in successfully served requests is considerable, which highlights the impact of OEA attacks on super-P2P systems.

We consider the MRT metric to show that malicious peers indeed make up a disproportional fraction of routing table entries for both passive and active OEAs. Indeed, for $MI = 10\%$, we observe an MRT between 18.5%-23.5% rather than the expected 10%, whereas $MI = 20\%$ increases MRT further to 38%-41.5%, as depicted in Figure 5.2b. Thus, malicious peers indeed manage to infiltrate routing tables.

5.6.4 Case Study 2: Detection assessment

In this study, we evaluate the reliability, accuracy and effectiveness of the proposed two-fold detection mechanism. We start by considering the composite proactive and reactive mechanism for various network sizes. Afterwards, we consider the proactive and reactive mechanism individually. For the first, we vary the number of correct replies l whereas we consider the impact of the fraction of routing table entries r that are considered untrustworthy on the latter. We compare the simulation results with our analytical results from Section 5.5.

Composite Mechanism Figure 6.3 exemplifies the effectiveness of our detection mechanism for $k = 12$, $l = 6$, $\tau = 100$ and $r = 0.1$. We vary the network size between 3,000, 6,000, and 10,000 peers. Malicious peers make up either $MI = 10\%$ or $MI = 20\%$ of the set of super peers.

Our detection scheme is indeed highly effective. The composite detection mechanism increases the lookup success ratio to at least 98% for all considered parameters, as depicted in Figure 5.3a. In comparison to the often enormous failure rate experienced in case study 1, we now can achieve success ratios close to the no-attack scenario.

We only incur a detection overhead of 4%-5% regardless of the network size, as illustrated in Figure 5.3b. The reason for the low overhead is that the mechanism only requires contacting a low fraction of peers in comparison to the flooding-based lookup mechanism. Furthermore, the mechanism is only applied at a low frequency, i.e., every $10s$ for the proactive and every $70s$ for the reactive mechanism. More frequent detection attempts did not considerably improve the performance.

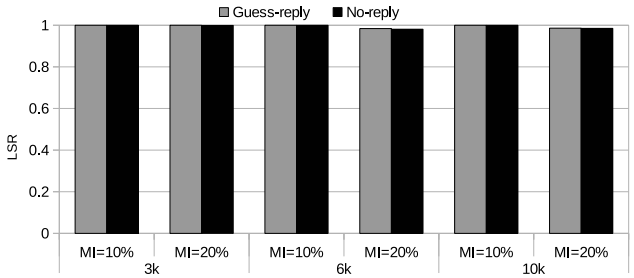
Figure 5.3c depicts the successful expulsion rate ERT, which provides a deeper understanding on how the detection mechanism works. Regardless of the attack strategy and infiltration rate, the expulsion ration ranges from 99.98%-100%.

Malicious peers are expelled locally via the proactive mechanism and globally via the reactive mechanism. Indeed, the result agrees with our analytical model, which states that the likelihood to expel malicious peers at some point is essentially 1. Similarly, both the simulation and the theoretical results (Equation 5.5) indicate that the chance to evict an honest peer is negligible. We consider the slightly higher chance of removing an honest node from a routing table in the following.

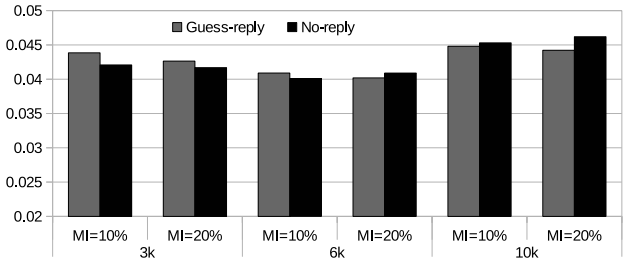
Proactive Mechanism We compare the theoretical and simulation results for the proactive scheme and elaborate on the impact of the parameter l. Throughout this evaluation of the proactive mechanism, we set $MI = 0.1$.

In general, our theoretical results for removing malicious peers from routing tables match closely with the simulation. The agreement between theory and simulation holds for both attack strategies: no-reply and guess-reply. For guess-reply, we restrict our results to the peers not contained in the guessed reply, as peers that are always contained in the reply are inherently unable to detect that the peer is malicious. Indeed, as peers are allowed to maintain θ neighbors, keeping these θ peers does not violate the protocol.

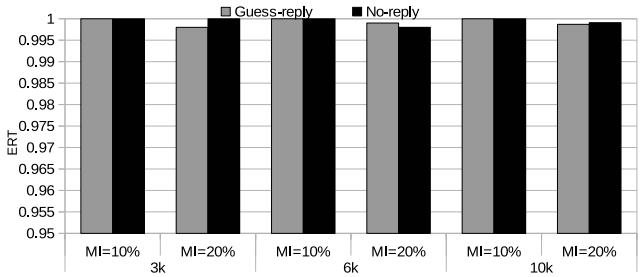
Figure 5.4a illustrates the closeness of the results for both guess-reply and no-reply with $k = 6$ and $l = 3$. We consider the cumulative distribution function



(a) Lookup success rate (LSR)



(b) Detection Overhead (DO)



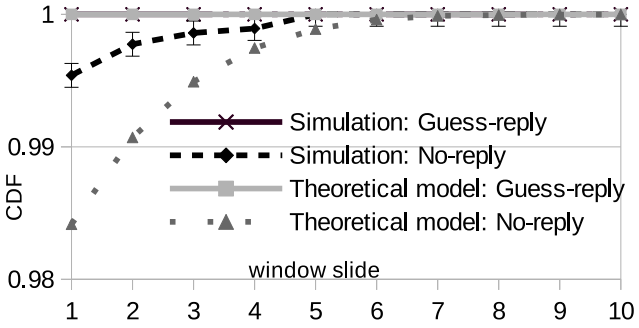
(c) Expulsion ratio per RT (ERT)

Figure 5.3: Detection mechanism performance

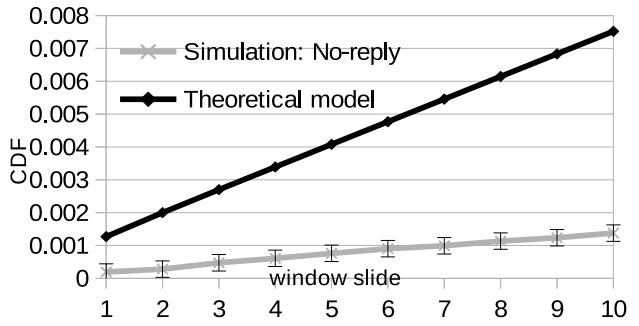
of the fraction of removed malicious peers in terms of the number of windows (i.e., the number of queried anonymizers minus (k-1)). If malicious peers send a random reply when asked by an honest anonymizer, peers that are not contained in the reply recognize the misbehavior almost immediately. Hence, honest peers almost always remove malicious peers in the first window if the malicious peer uses guess-reply as a strategy, resulting in a removal rate of 1 in Figure 5.4a.

In contrast, if the malicious peer chooses not to reply, it might initially not be detected if there are at least l malicious anonymizers. However, the likelihood to continuously choose such l anonymizers decreases exponentially over the number of windows. Hence, malicious peers applying no-reply are detected rapidly as well, as depicted in Figure 5.4a.

We consider the honest removals in Figure 5.4b. The theoretical model presents an upper bound on the cumulative removal rates exhibited in the simulation. Overall, the chance of removing an honest peer from a routing table



(a) Malicious-removals CDF



(b) Honest-removal CDF

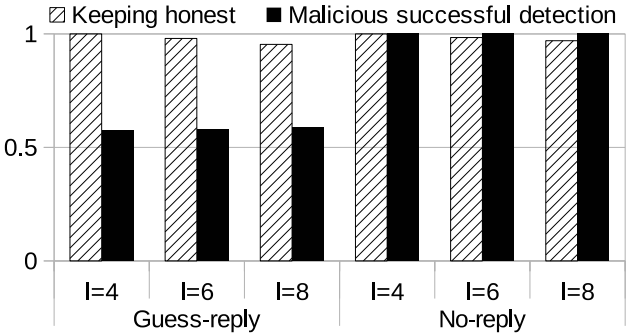
Figure 5.4: Comparing theoretical and simulation results $(k, l) = (6, 3)$

is several orders of magnitude lower than for a malicious peer. Hence, our proactive mechanism rarely disadvantages honest peers.

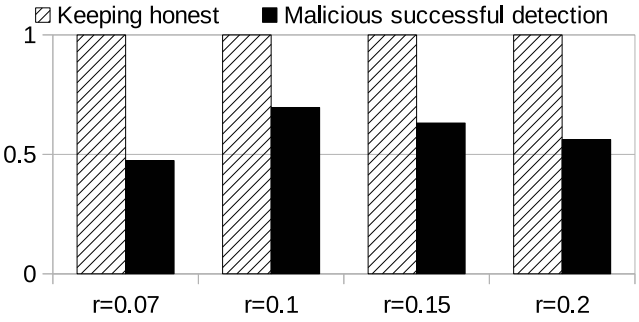
Now, we evaluate the effect of the number of correct replies l. For that purpose, we fix $k = 12$ and choose $l \in \{4, 6, 8\}$. As shown in Figure 5.5a, for both malicious behaviors types (guess-reply and no-reply), the honest keeping ratio HRT exhibits a slight decrease when l increases. The reason is the increased number of required honest anonymizers, which are increasingly unlikely to have over an extended period of time. In contrast, even at low values of l, the malicious detection rate is not equally affected as it is already at its maximum.

Reactive Mechanism In this study, we assess the reactive mechanism, we vary $r \in \{0.07, 0.1, 0.15, 0.2\}$. We set $\tau = 100s$ and $MI = 10\%$. The effect of varying r is shown in Figure 5.5b. Choosing a very small value $r = 0.07$ decreases the probability of successfully expelling malicious peers. The lack of successful detection is due to that a specific malicious peer is unlikely to rank lowest in the routing table of the majority of its neighbors, as these might have multiple malicious peers. However, at $r = 0.1$, the ratio of successfully detected malicious peers increases to 70% as a result of including more malicious peers in $L_{w,r}$.

Note that at higher values $r = 0.15, r = 0.2$, honest peers are also included in $L_{w,r}$ by some of their neighbors. Nevertheless, the overall eviction rate of honest peers only increases slightly because honest peers are unlikely to be



(a) Proactive: varying l



(b) Reactive: varying r

Figure 5.5: Evaluating parameters for detection mechanism

declared untrustworthy by all peers. To that end, we highlight the effectiveness of our detection mechanism for a wide range of parameters.

5.7 Conclusion

In this work, we highlight the severity of launching OEAs against super-P2P unstructured overlays. Consequently, we propose a two-fold detection mechanism that constitutes of a proactive and reactive mechanisms.

Through an analytical and simulation-based evaluation, the proposed mechanism restores the overlay's reliability up to 100%, achieves 99% successful malicious detections, and entails a low network overhead (4%-5%). Moreover, we note that an interesting carry on to this work is to combine the proposed algorithms with resilient online streaming P2P networks. Next, a general conclusion to the whole thesis is described in Chapter 7 which emphasizes the highlights and the results in each research question along with the corresponding contribution(s).

Chapter 6

Streaming Overlay: Defending Against Cheating Attacks

This chapter address Research Question 3 and the subsequent Contribution 4 published in [IRS18b]. We address the resiliency of online streaming overlays to a DoS attack variant. Specifically, Internal buffer-Map BM cheating attacks are being investigated, where *internal* refers to the capability of the attacker to actually insert malicious peers in the system, rather than only being able to *externally* shut down resources. Accordingly, a detection mechanism that guarantees peers satisfaction are rapidly restored is presented and evaluated theoretically and through simulations, while the proposed cheating attack severity is assessed as well.

The work is presented as follows: Section 6.6 discusses the related work and the background. In Section 6.2, the concepts underlying the attacker model internal attack and the conducted adversarial behaviors are defined. The detection mechanism is described in details in Section 6.3, with the theoretical analysis is discussed in Section 6.4. The internal attack's impact, along with the detection mechanism performance and efficiency, are evaluated in Section 6.5. Finally, we conclude our work and discuss the future work in Section 6.7.

6.1 Motivation: Streaming & Cheating Attacks

Streaming content is an essential component of data-driven infrastructures [Tha13]. Most streaming applications rely on (monopolistic) central providers that have significant computing and communication resources to distribute high quality of service content to a large audience in a fast and reliable manner. For the providers, this entails the use of dedicated resources, involves performance and scalability issues along with considerations of handling single point of failures. Distributing content via a Peer-to-Peer (P2P) network significantly lowers the load that the provider experiences as the participants relay the downloaded content to other participants. This eliminates the need that all participants receive their content directly from the source, i.e., the provider. Consequentially, P2P

streaming becomes an attractive option for alternative distributed providers and user-generated content to help reduce data delivery costs and results in lower rates for customers.

However, achieving high reliability in a P2P overlay and across a dynamic and heterogeneous group of content distributors is challenging. In addition to the inherent operational unreliability of benign participants, attackers such as competitors can infiltrate the set of peers and conduct denial-of-service attacks. In this manner, malicious participants can interrupt or delay the distribution of the content with the goal of degrading the quality of service.

In general, nodes in a P2P streaming system connect to a small set of other nodes, called the *neighbor set*. The publisher or the *source* of the content divides the stream into *chunks*, each contains an equal-sized part of the encoded data, and forwards these chunks to its neighbors. Nodes in the system receive chunks from neighbors and forward them to neighbors that have not previously received the respective chunks. The selection of neighbors and the choice of neighbors to receive-from or forward-to differs across protocols [LGL08]. Yet, pull-based mesh networks are the predominant method in P2P streaming systems [ZXW+14]. In a mesh overlay, peers maintain a buffer-map indicating which chunks they possess. Neighbors periodically exchange their buffer-maps and request chunks from neighbors whose buffer-maps indicate possession of the respective chunk. Peers then forward chunks based on the received requests.

In pull-based systems, there exist several denial-of-service attacks, known as buffer-map or BM cheating attacks [CLW07]. In such an attack, malicious nodes might drop or delay chunks. Alternatively, they might advertise chunks that they do not have. As detection of the latter is locally possible and the effect of delaying chunks is at most as severe as entirely dropping the respective chunks, we focus on a denial-of-service attack through dropping chunks without advertising them, called *Drop-chunk* in the following.

In the past, multiple countermeasures aimed to reduce the severity of the *Drop-chunk* attack. However, the majority of these defences [ZLL+05; HC04; SR12] assume that the attacker is unaware of the topology of the streaming network and specifically does not know the *headnodes*, i.e., the peers connected to the source. However, previous work indicates that it is relatively easy to infer the identity of benign headnodes and then target those important nodes [NRS+16]. While countermeasures to these inference attacks exist, they assume an external adversary that can shut-down or replace certain nodes [NRS+16; NF15; NFS14]. Hence, the existing work evaluates neither the impact of internal attacks nor defending against internal attackers, i.e., attackers that insert nodes under their control into the system pretending to be regular participants.

6.1.1 Contributions:

In this work, we first illustrate the effectiveness of internal attacks based on malicious headnodes. Consequently, we propose a mechanism for detecting *Drop-chunk* by keeping track of peer satisfaction. If the cumulative satisfaction level of a group of peers drops below a certain threshold, the source replaces all headnodes associated with the group with randomly chosen peers. Hence, our detection and mitigation mechanism efficiently reacts to a detected low quality of service rather than explicitly identifying the misbehavior of one or more specific nodes. Note that we focus on attackers that control a low fraction of nodes,

as attackers controlling the majority of nodes can trivially control most of the communication, even without gaining strategic positions such as headnodes.

Using a simulation study, we show that the proposed detection mechanism effectively restores peers satisfaction, even in a scenario where the attacker controls all headnodes. Our mechanism introduces only a small signaling overhead of approximately 8% supporting the claim of being efficient.

A theoretical analysis complements our practical results, focusing on the opportunities to abuse the detection mechanism. Indeed, the detection algorithm prevents malicious nodes from replacing benign headnodes with malicious nodes unless they control a large fraction of the total number of nodes. Furthermore, maintaining malicious headnodes despite generally low satisfaction levels is not possible for the considered attacker.

6.2 Internal Attack Model

In this section, we introduce the concept of internal attacks in streaming P2P overlays. Our focus is on attacks on availability that aim to intercept data chunks from the source. We start by introducing the attack characteristics such as target, budget, and malicious nodes placement. Subsequently, our main discussion outlines the *Drop-chunk* adversarial behavior.

6.2.1 Target, budget and placement

The target of the internal attack is to severely degrade the user's satisfaction by interrupting the stream close to the source, thus preventing dissemination between benign peers. The budget x of the attacker corresponds to the number of nodes controlled by the malicious party. In accordance with the attack goal of maximizing impact, the attacker aims to use its budget to occupy the source's neighbor list. Note that in a real streaming system, the typical size of the source's neighbor list is 20-30 entries [HFC+08; VGL+07], which highlights the feasibility of conducting an internal attack using a very small budget.

We assume an attacker to (a) have a budget x, and (b) be capable of assigning malicious peers as headnodes. Potential ways of assigning headnodes include (1) joining the overlay as early as possible in case of a pre-announced time for a streamline, (2) taking down the source's benign headnodes, or (3) abusing peers' replacement mechanisms [NRS+16]. Hence, the attacker initially assigns $x_h \leq x$ of its resources as headnodes. As the attacker's main objective is to fully occupy the source's neighbor list, the optimum value of x_h for the attacker, is $x_h = |NeighborList|$. If full exploitation of the source's neighbor list is not feasible when the attack is being initiated, the attacker continuously tries to increase the value of x_h.

The rest of malicious peers $x - x_h$ are connected as neighbors to the x_h headnodes. Such a placement is the best strategy for the attacker since the impact caused by the $x - x_h$ peers is maximized due to their relative closeness to the source, i.e., a larger fraction of benign peers experience a longer service degradation till a sufficient amount of benign peers receive and start disseminating the stream. Given the fact that inferring the overlay's topology is indeed feasible [NRS+16; NF15], the attacker is capable of inferring the existing headnodes to optimally place malicious peers as headnodes. Knowing the

headnodes allows the attacker to place malicious nodes in their neighborhood, which also results in an effective disruption of the stream dissemination if the budget is sufficiently high.

6.2.2 *Drop-chunk* adversarial behavior

We now discuss the main adversarial behavior that gets executed based on the attacker's target and budget. Let M be the set of malicious peers that collaboratively execute *Drop-chunk*. When $m \in M$ receives a chunk from a neighbor, m drops the chunk. In particular, m never advertises chunks in its BM, except to the neighbor it received the chunk from. Indeed, it keeps on requesting the dropped chunks from other benign peers $b \in B$, where B is the set of benign peers. This scenario guarantees that: (a) malicious peers are less susceptible to being suspected as the requesting benign peers are not aware that m indeed received those chunks, and (b) detecting m's direct or close connection to the source, inferring the overlay's topology, is not possible, which lowers the probability of m being suspected.

Note that this behavior minimizes the detection susceptibility of malicious peers. The reason is that other BM cheating strategies result in eventually declaring a certain suspect, e.g, if m keeps on sending correct BM updates but never sends the actual chunk, honest nodes will eventually suspect m.

6.3 Detection Mechanism

In this section, we explain our detection mechanism, starting with an overview of the different steps followed by a detailed description of each step. The goal of the detection algorithm is to restore the user's satisfaction in the face of a *Drop-chunk* attack. The key idea of our method is to replace headnodes associated with peer groups of low satisfaction levels.

Throughout the section, we assume that nodes authenticate their messages using digital signatures. The source keeps track of participants' verification keys and can hence establish the authenticity of messages. In particular, malicious nodes cannot forge responses of honest peers to influence the mechanism. We assume that neighboring nodes periodically exchange messages stating that they are neighbors. These *proofs of neighborhood* are signed and contain a time stamp. In this manner, u can proof if v is (or has recently been) its neighbor.

6.3.1 Mechanism Overview

We illustrate the underlying ideas of the detection mechanism in Figure 6.1. When a malicious peer m performs a *Drop-chunk* attack, benign peers b are unable to immediately identify the malicious behavior. Specifically, m never sends the actual BM that represents the chunks it currently possesses, i.e., m is only requesting chunks it already has. Thus, detecting a violation in this case is not straight-forward. In particular, nodes are generally unable to identify a suspected attacker based only on local information. In the remainder of the section, we present a mechanism that allows nodes to collaboratively identify suspects that are subsequently removed as headnodes.

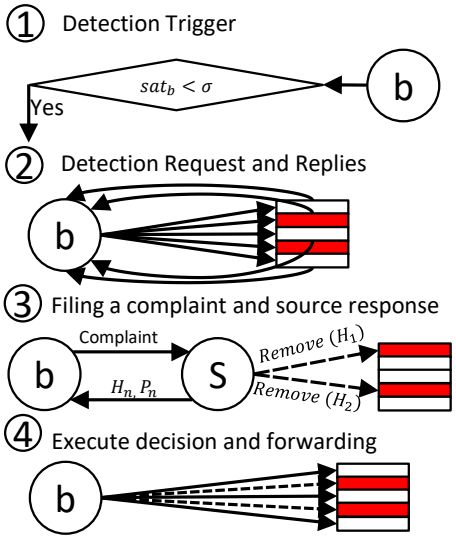


Figure 6.1: Detection process for *Drop-chunk*. S denotes the source.

The detection consists of four steps, starting with an initial trigger of dissatisfaction at one peer and potentially terminating in replacing one or several headnodes. First, when a peer b suspects a *Drop-chunk* attack based on its local observations, b sends a *detection request* to all peers in its neighbor list. Second, each peer receiving a detection request prepares a response. Third, the initiator b decides based on the received responses if they should file a complaint to the source. If b decides to file the complaint, b sends it on behalf of the participating peers in the request. Afterwards, the source verifies the complaint, reacts accordingly, and responds to b, detailing the steps taken. The reaction of the source is either the replacement of one or several headnodes or the rejection of the removal request. Finally, b reacts based on the received response from the source and then forwards the source's reply to the other participants in the complaint, who in turn execute the same procedure.

The node b bases its decision on whether to initiate a request or forward a complaint on a number of threshold parameters, which we summarize in Table 6.1 together with various system parameters governing the attack.

Table 6.1: Detecting cheating attacks: acronyms description

Var.	Description	Var.	Description
H_n	headnodes in a complaint	P_n	potential candidates
x	no. of malicious peers	x_h	mal. headnodes
BM	buffer-map	$x - x_h$	mal. non-headnodes
sat	peer satisfaction level	σ	satisfaction threshold
t_{drop}	min. drop responses	κ	no. allowed det.

6.3.2 Detection Trigger

In order to start a detection request, the node b has to experience a low satisfaction level. The satisfaction of a peer is defined as the fraction of missed chunks, i.e., the continuity of the stream according to the $Hit/Hit + miss$ chunk ratio. In a nutshell, b starts a detection request if its satisfaction is below a satisfaction threshold σ. However, to limit the ability of malicious peers to incorrectly accuse benign peers and increase the load through false detection requests, the concrete conditions that result in a detection request from b are:

1. b's current satisfaction level sat_b is less than the predefined threshold, i.e., $sat_b < \sigma$.
2. The number of drop detection requests sent by b in the time interval t_{det} is less than κ.
3. b has not initiated or replied to any other *Drop-chunk* detection request that the source has not decided on yet.

The second and third condition guarantee that peers cannot abuse the mechanism via triggering or participating in multiple detection requests in parallel. Moreover, restricting concurrent requests for benign peers is sensible as their low satisfaction level is already noted in their reply to previous requests.

6.3.3 Processing a Detection Request

Let D denote the set of queried peers, i.e., the neighbors of the initiator b if b executes the protocol honestly. When receiving a detection request, a peer $d \in D$ hence first checks if it can participate in any more requests. If so, d replies with its sat_d and a time stamp, both signed by its private signature key. The time stamp prevents the attacker from replaying benign peers' previous (low) satisfaction levels, as only recent satisfaction levels are valid.

6.3.4 Filing and Processing a Complaint

Upon receiving a feedback from its neighbors, b decides whether to file a complaint or not. If so, the source verifies the complaint and potentially removes some of its headnodes.

Filing a complaint

b will start processing the replies once all nodes in D have replied or a time-out $t_{replies}$ occurs. We assume that the source's address is publicly known and b can send a complaint to the source directly.

b sends a complaint if the average satisfaction level indicated in the responses is below a threshold σ and at least t_{drop} peers replied to the request. More specifically, let $sat_1, \ldots, sat_z$ be the satisfaction levels expressed in the replies and sat_b be b's satisfaction level. Assuming a sufficient number of replies, b files a complaint to the source if:

$$\frac{1}{z+1}\left(sat_b + \sum_{i=1}^{z} sat_i\right) < \sigma. \tag{6.1}$$

Otherwise, b either starts another detection request depending on κ or waits until allowed to send another detection request.

Once b decides on filing a complaint according to the aforementioned conditions, b generates a complaint message to the source containing the IDs of all nodes in D, recent proofs of neighborhood, and the received satisfaction levels including signatures and time stamps.

The reason for requesting at least t_{drop} replies is to prevent a few malicious nodes from accusing benign headnodes. By imposing a lower bound on the requested number of replies, a considerable number of malicious nodes has to use one of their κ requests. We present an in-depth analysis on how these constraints prevent misuse in Section 6.4.

Processing a Complaint at the Source

The source s first verifies the content of the complaint. First, the source rejects any complaint from a node b that has already participated in κ requests. If s does not reject the complaint, s then removes any satisfaction levels without valid signatures from the complaint. Furthermore, s removes any responses from nodes that have exceeded their participation limit or are participating in two complaints at the same time.

If the remaining valid responses still indicate an average satisfaction level of less than σ, the source:

1. divides the set of peers in D into two sets H_n and P_n, where H_n is the set of headnodes peers that exist in the complaint.
2. removes all peers in H_n from its neighbor set.
3. randomly connects to another $|H_n|$ peers.
4. adds peers (excluding peers in H_n) from its neighbor list to P_n, where $P_n = NeighborList \setminus H_n$ ($NeighborList$ is the set of peers in a neighbor list).
5. sends a *Complaint Reply* to b containing H_n and P_n.

The reason for choosing random new headnodes rather than nodes participating in the complaint is to lower the incentive for complaints by malicious peers. Even if such a complaint is successful, the new headnodes are likely benign, meaning that the malicious nodes did not gain anything from initiating the request apart from slightly increasing the load.

Processing a Complaint Reply & Forwarding

Finally, when b receives the *complaint Reply* from the source, b

1. Disconnects from all peers in H_n. Note that b does not blacklist peers in H_n from its neighbor list due to the fact that those peers are not proven malicious.
2. Connects to $|H_n|$ peers from P_n, in case $|H_n| > |P_n|$, peers connect to $|P_n|+(|H_n|-|P_n|)$ random peers.

Subsequently, b forwards the complaint to the other participants, i.e., $D \setminus H_n$, who in turn execute steps 1 and 2.

6.3.5 General Notes

The detection mechanism does not aim at expelling peers from the system. Simply removing headnodes remarkably benefit the system. Indeed, the only peers that get blacklisted are those who violate the detection mechanism constraints, i.e., participating in more than a single request at a time or initiating more than κ requests. The reason for this leniency lies in the potentially high chance of removing headnodes that are benign but exhibit a low performance. In general, the main target of the detection mechanism is to enhance peers' satisfaction level while keeping peer replacements and signaling overhead low.

6.4 Analysis

We focus on characterizing the behavior of malicious nodes aiming to subvert the detection mechanism to remove honest headnodes and retain malicious ones. More precisely, we show that successfully accusing a benign headnode of cheating requires that the malicious peer issuing a complaint presents a neighbor list that is either dominated by malicious peers or by benign peers with unusually low satisfaction levels. Similarly, preventing the removal of a malicious headnode requires that a high number of the neighbors are malicious.

6.4.1 Falsely Accusing Benign Headnodes

We start by considering the case that malicious nodes want to misuse the detection mechanism to remove a benign headnode. Note that there are reliable methods to identify headnodes [NRS+16], so malicious peers are likely to know if one of their neighbors is a headnode. The malicious node m initiating a request with the goal of removing one benign headnode can manipulate the set D of nodes m forwards to the source. In other words, after querying all nodes in its actual neighbor list, m might send only subset of the responses as well as responses from additional nodes to the source. If possible, m chooses these responses in such a manner that the source will remove the benign headnode. There are restrictions guiding the construction of D that m has to take into consideration:

- m should include the benign headnode it aims to remove.
- m cannot include benign nodes that are not in its actual neighbor list, as m has no valid proofs of neighborhood.
- m does not have to include all peers that are in its actual neighbor list, as there is no possibility to detect excluded neighbors short of asking all peers in the system if they are neighbors of m.
- m can include malicious peers that are not in its actual neighbor list, as these peers are willing to generate false proofs of neighborhood. Only the inclusion of malicious nodes that can participate in a *Drop-chunk* request, i.e., those that have not yet reached their limit of *Drop-chunk* request participation, is beneficial for the success of the request. Malicious peers contained in D claim that their satisfaction level is 0 to maximize the chance of removal.

When deciding on a set D, m tries to minimize the number of malicious nodes in D in order to use as few of the κ requests per node as possible. Preposition 6.4.1 gives a lower bound on the number of required malicious nodes.

Preposition 6.4.1. *Let m be a malicious neighbor of a benign headnode h with satisfaction level sat_h. Assume that m has k benign neighbors $v_1, \ldots, v_k$ sorted by their satisfaction levels $sat_1 \leq sat_2 \leq \ldots \leq sat_k$. Then, to successfully remove h, m has to include at least c responses of malicious nodes, including m itself, in the set D of forwarded responses such that:*

$$c = \max\Big(1, \quad argmin_{c' in \mathbb{N}} \left\{ \frac{1}{t_{drop}} \left(sat_h + \sum_{i=1}^{t_{drop}-c'-1} sat_i \right) < \sigma \right\}\Big) \tag{6.2}$$

Proof. To remove a headnode h, there has to be a detection request containing the responses of h and $n \geq t_{drop} - 1$ nodes with satisfaction levels $s_1, \ldots, s_n$ and $\frac{1}{n+1}\left(s_h + \sum_{i=1}^{n} s_i\right) < \sigma$. The node m aims to minimize the number of involved malicious nodes c because each malicious node can only participate in κ detection requests per interval. At the same time, m has to ensure that the average satisfaction level of the involved nodes is below σ and that the request includes at least t_{drop} nodes in total. As m files the request, at least one malicious node has to be included. In other words, m solves the optimization problem of finding a minimal c and a set of integers $I \subset \{1, \ldots, k\}$ such that i) $c + |I| + 1 \geq t_{drop}$, ii) $\frac{1}{c+|I|+1}\left(sat_h + \sum_{i \in I} sat_i\right) < \sigma$, and iii) $c \geq 1$. Choosing the lowest satisfaction levels indeed solves the optimization problem and results in Eq. 6.2. □

For simplicity, Preposition 6.4.1 considers the case that only one headnode is contained in m's neighbor list. In the presence of several headnodes, m has to slightly adapt its attack strategy. If additional malicious headnodes are neighbors of m, m does not include the respective nodes in D to avoid accidentally causing the removal of malicious headnodes. In contrast, if additional benign headnodes are neighbors of m, m will include all of them in D if the detection request can be successful. If success is not possible due to the high satisfaction level of the included headnodes, m successively removes each headnode using the strategy outlined in Preposition 6.4.1.

6.4.2 Retaining Malicious Headnodes

Now, we consider the case that malicious nodes collude to retain one or several malicious headnodes when a benign peer initiates a detection request. All malicious nodes in the respective neighbor list will provide a satisfaction level of 1 to prevent the removal of a malicious node. We assume that malicious neighbors will try to prevent the removal of malicious headnodes even if the request can additionally result in the removal of benign headnodes. This assumption seems reasonable as the removal of a benign headnode is unlikely to lead to additional malicious headnodes, indicating that retaining existing malicious headnodes is of higher importance than removing benign headnodes. Preposition 6.4.2 provides the condition governing the success or failure of the *Drop-chunk* request in the face of the proposed adversarial behavior.

Preposition 6.4.2. *Let m be a malicious headnode and b be a benign neighbor of m that initiates a detection request due to its low satisfaction level sat_b. Assume that b has k benign neighbors $v_1, \ldots, v_k$ with satisfaction levels $sat_1, sat_2, \ldots, sat_k$. In addition, b has y malicious neighbors, which includes the malicious headnode, and $k \geq t_{drop}$. Then the removal of m fails if and only if:*

$$\frac{1}{k+y+1}\left(y + sat_b + \sum_{i=1}^{k} sat_i\right) \geq \sigma. \tag{6.3}$$

Proof. The claim follows directly as all y malicious peers will set their satisfaction level to 1 and *Drop-chunk* requests with an average satisfaction of at least σ are not successful. □

6.5 Evaluation

The goal of this section is to address two research questions: First, we quantify the severity of *Drop-chunk*. Second, we evaluate the proposed detection mechanism's performance in terms of effectiveness and efficiency. We start by describing the simulation model and set-up before detailing the simulation results and their interpretation for both research questions.

6.5.1 Simulation Framework, Parameters and Metrics

Our simulation framework builds on OSSim [NFS13]. OSSim is a packet level simulator for DONet [ZLL+05], a pull-based online streaming overlay. All our overlays use the network topology generator GT-ITM [ZCB96] with 1000 peers connected to 400 edge router. Furthermore, our simulation time is 500s and the presented results are averaged over 10 runs.

We differentiate between malicious and benign peers when considering their online times. We assume that malicious peers join the overlay early and do not leave before the content dissemination ends in order to maximize their impact. In contrast, benign peers join based on Pareto distribution and leave according to Lognormal distributions, as motivated by real-world measurements [VAM+06]. Benign peers can rejoin the overlay in a uniform distribution around 10s. For both case studies, the streaming rate is $400kbps$, the chunk size $2500B$ and the buffer size $30s$.

The following metrics characterize the performance.

Satisfaction *sat*

The satisfaction is the fraction $Hit/Hit + miss$ of chunks peers receive in time, averaged over all peers.

Avg. Loss *lo*

Indicates the average number of chunks that peers do not receive in time, averaged over all peers.

Detection Overhead *DO*

The detection overhead describes the communication overhead created by the detection mechanism. More formally, it is the ratio of messages exchanged in the system due to the detection mechanism and all signaling messages: (1) *BM* requests, (2) *BM* updates, and (3) neighboring requests, accepts and rejects.

Benign Ratio per Neighbor List *BRNL*

The benign ratio per neighbor list measures the fraction of benign peers in the source's neighbor list.

6.5.2 Case 1: *Drop-chunk* Severity

In this case study, we evaluate the impact of *Drop-chunk* on two different network scenarios: (1) DONet, and (2) DONet+SWAP [NRS+16]. We consider SWAP to check how replacement mechanisms of headnodes affects the attack. We use the same total number of peers but vary the attacker's budget. As malicious peers aim to occupy the closest peers to the source, the remaining size of the overlay is not a factor on the impact of the *Drop-chunk* attack.

Given the source's neighbor list size $LS = 10$, we choose the following combinations for the attackers budget $(x_h, x - x_h)$: $(10, 0), (5, 15), (7, 49), (8, 24)$. Here, $x - x_h = 49$ denotes that 7 malicious peers are connected to each of the 7 malicious headnodes. First, we analyze the attack's impact on DONet and then we evaluate the resilience of SWAP to the attack.

Figure 6.2a displays the average chunk loss ratio. Unsurprisingly, the average loss is 100% when $x_h = LS = 10$, which means that the source's LS is utterly saturated with malicious headnodes, i.e., no chunks are transmitted to the rest of the overlay. Thus, the average peer satisfaction is always 0%.

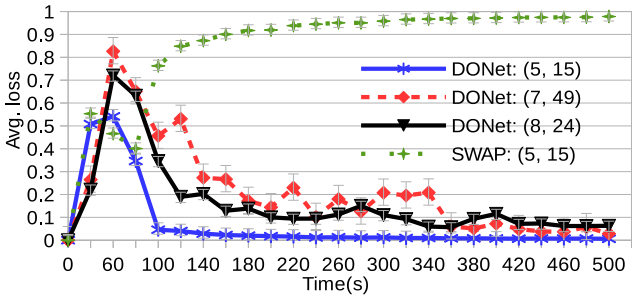
If $x_h < 10$, the average loss initially reaches up to 82% for $(x_h, x - x_h) = (7, 49)$, for $(x_h, x - x_h) = (5, 15)$, the loss ratio is 54% and 73% for $(x_h, x - x_h) = (8, 24)$, as shown in Figure 6.2a. If x_h or $x - x_h$ increases, benign peers experience severe service degradation for a longer time period. Benign peers close to the source suffer from overload, leading to a high ratio of missed chunks. Nevertheless, the loss ratio decreases once a fraction of benign peers are able to serve the rest of the overlay.

Figure 6.2b presents the average peer satisfaction level *sat*. As a consequence of experiencing high chunk loss rate, higher values of x_h and $x - x_h$ result in lower peer satisfaction over time, where benign peers at $(x_h, x - x_h) = (5, 15)$ restore their satisfaction level at approximately 340s, which is earlier than at $(x_h, x - x_h) = (7, 49)$ and $(x_h, x - x_h) = (8, 24)$.

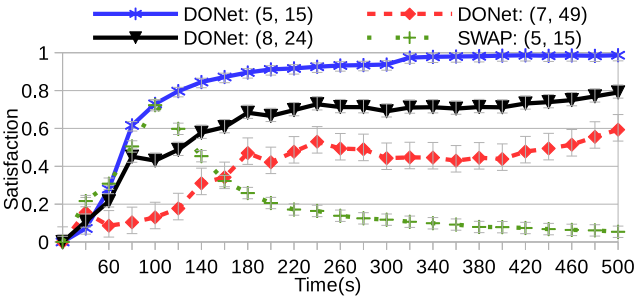
Now, we analyze the attack's impact while SWAP is operating. During SWAP, peers nominate new headnodes and forward these nominations to the source. Malicious peers abuse the mechanism by nominating other malicious peers at each nomination round. Moreover, malicious peers connected to benign headnodes are eventually nominated to the source and thus can occupy the source's neighbor list LS.

As shown in Figure 6.2a, comparing the same values at $(x_h, x - x_h) = (5, 15)$ for both DONet and SWAP show that the impact of the attack is more significant if SWAP is active. Before the source's LS is saturated with malicious

peers at $t = 80s$, the average loss is in fact decreasing, However, as soon as malicious peers control the neighbor list, the average loss increases up to 97%. For the same reason, the satisfaction level of benign peers eventually decreases to 6%, which highlights the unsuitability of SWAP against our proposed attack.



(a) Avg. loss



(b) Avg. peer satisfaction

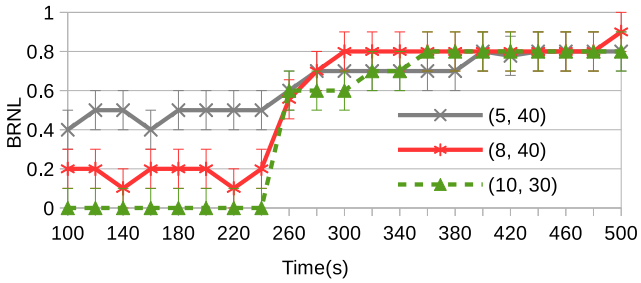
Figure 6.2: Attack's impact on DONet

6.5.3 Case 2: Detection Mechanism Performance

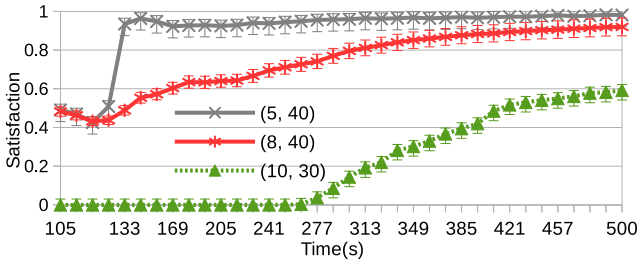
We now evaluate the performance of the detection mechanism. Benign peers execute the detection mechanism as described in Section 6.3 whereas malicious peers aim to misuse the mechanism. More precisely, malicious peers reply with a satisfaction level of 0 if the complaint might remove a benign peer and 1 if it might remove a malicious peer.

In order to do so, we chose $(x_h, x - x_h) \in \{(5, 40), (8, 40), (10, 30)\}$ to assess the performance of the mechanism in severe attack conditions. The satisfaction threshold σ is set to 0.95 to measure if peers are able to fully restore their satisfaction level when the detection mechanism is operating. The detection mechanism is effective starting $t = 250s$ to allow for a reasonable amount of peers to join the overlay to adequately assess the efficiency of the mechanism. In this scenario, every peer is allowed to initiate $\kappa = 10$ detection requests for $t_{det} = 500s$, and the minimum number of responses to generate a complaint is $t_{drop} = 3$. We discuss the effect of varying those parameters later on.

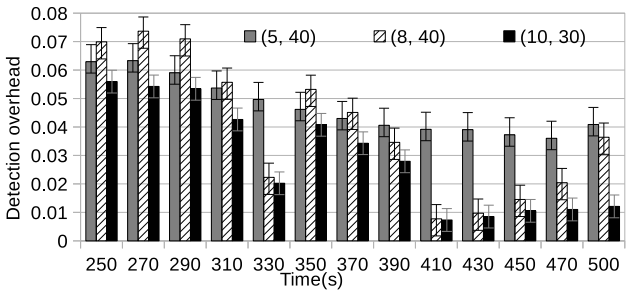
As depicted in Figure 6.3a, we observe an increase in the benign headnodes ratio in the source's neighbor list after the detection mechanism starts operating



(a) Average BRNL



(b) Average peer satisfaction



(c) Detection overhead

Figure 6.3: Detection mechanism performance

at $t = 250s$. For instance, if $(x_h, x - x_h) = (5, 40)$, the source successfully attains 80% benign headnodes due to the detection mechanism. For $(x_h, x - x_h) = (8, 40)$, the $BRNL$ ratio increases up to 90-100%, which reflects the efficiency of the detection mechanism in replacing malicious headnodes to restore peers' satisfaction levels. Even if the source is initially only connected to malicious headnodes, i.e., $x_h = 10$, the detection mechanism is capable of restoring a $BRNL$ to approximately 80%.

Figure 6.3b illustrates the average restored peer satisfaction level when the detection mechanism is active. For all considered attack budgets, the average satisfaction level quickly increases to 95-100% for $(x_h, x - x_h) = (5, 40), (8, 40)$. For $x_h = 10$, the average satisfaction increase from 0% to almost 60% in a time span of 250s. The reason of the quick increase is that the number of initial detection requests sent to the source results in replacing a high fraction

of the source's headnodes. Moreover, malicious peers are unable to misuse the mechanism, as indicated by the absence of degradation in satisfaction levels.

Figure 6.3c depicts the average detection overhead induced by our mechanism. The maximum overhead due to the detection mechanism is 8% for all scenarios. As peers are eventually satisfied, the number of detection requests initiated decreases and the overhead decreases to 4% at $t = 500s$, i.e., peers stop invoking the mechanism. Moreover, the maximum number of detection requests that can be initiated is dependent on κ, which is set to 10 in this scenario. Thus, smaller values of κ result in lower overhead. In fact, varying t_{drop} between 3, 4 and 5 has little impact on the detection performance, indicating that nodes receive sufficient replies.

In summary, our simulation study highlights the effectiveness efficiency of the detection mechanism against *Drop-chunk* attacks, even in the presence of an attacker that initially controls the majority of nodes close to the source.

6.6 Related Work

We overview the prominent existing work on attacks and their detection in the context of P2P streaming systems. Most prior work has considered three attack types: (i) pollution attacks, i.e., flooding the overlay with arbitrary content and claiming it to be relevant chunks, (ii) free riding, i.e., participating in the overlay without contributing, and iii) cheating attacks, i.e., maliciously dropping packets or manipulating buffer-maps.

Pollution attacks are considered one of the most common attacks [GD16]. Various mitigation and detection strategies exist a such as network coding [FGG15] and [KW14] that can effectively mitigate these attacks.

In contrast, the main approach to counter free riding are incentives [LWC09; HC04], i.e., rewarding peers that distribute the stream to others. However, these strategies are only effective for peers that aim to minimize their level of participating.

Cheating attacks are severe DoS attacks, performed to maximize the damage to the overlay and preventing peers from downloading the stream. *Antiliar* is a general defense mechanism against a diverse set of attacks, including dropping and buffer map manipulation [SR12]. Mainly, *Antiliar* tracks peers behaviors in a secure progress log and thus, detecting misbehaving peers by identifying irregularities in the log. While highly effective, *Antiliar* relies on expensive cryptographic operations that are unsuitable for devices with low CPU resources. Moreover, *Antiliar* uses a central entity to review the logs, creating additional security and privacy problems.

An alternative decentralized approach [NFS14] relies on redundancy by enforcing diversity when requesting chunks. In this manner, the attacker has to control a higher fraction of nodes to achieve any severe damage by cheating. The work focuses on attacks on headnodes yet assumes an external attacker that can take over arbitrary nodes at will. In this context, the idea of swapping headnodes frequently to mitigate the impact of the attacker's control significantly decrease the attack severity [NRS+16]. As shown in Section 6.5, internal attackers can undermine the swapping protocol and gain the position of headnodes.

6.7 Conclusion & Future Work

In this work, we focus on the class of internal inference attacks for pull-based overlays. The attacker conducts a BM cheating attack after placing malicious peers as headnodes. We show that the attack severity significantly increases the chunk loss ratio, accompanied by low satisfaction level experienced by benign peers. As a countermeasure, we propose a detection mechanism where peers are able to collaboratively file a complaint to the source when their average aggregated satisfaction drops below a certain threshold so the source can replace suspicious headnodes.

Our simulations show that the detection mechanism is capable of restoring 95-100% of peers satisfaction level while removing 80-90% of malicious headnodes from its neighbor list while inducing a low overhead of approximately 8%. As an ongoing work, we focus on evaluating the resilience of our approach against various BM cheating strategies and integrating anonymous monitoring for proactive defense. We note that we provide the main conclusion of the thesis, w.r.t. the research questions and contributions discussed in each chapter in the thesis in Chapter 7.

Chapter 7

Summary and Conclusions

Throughout the work conducted in this thesis, we investigated a critical question: ***"how secure are P2P systems?"***. Our primary focus was:

> *(1) Evaluating the feasibility and severity of conducting progressive attacks on different P2P overlays when a specific overlay-based vulnerability is exploited.*
>
> *(2) Designing an effective, accurate and lightweight distributed detection and mitigation scheme that is suitable to the constraints of each investigated overlay.*

As aforementioned in Section 1.1, In this thesis, we comprehensively investigate the security threats of structured, unstructured and streaming overlays. We targeted existing and progressive attacks, i.e., mutating attacks via exploring the feasibility of conducting advanced, more severe and combined adversarial behaviors when considering an attack form on a designated overlay.

In more details, through the rest of this chapter, we start by providing a high-level conclusion to the work conducted in this thesis. Afterwards, a detailed summary covering the main concepts and results of each research question with the corresponding contribution(s) is provided.

7.1 Attacks on P2P Systems

In order to cover different security exploitations that threaten P2P overlays, we focused on varied threat models throughout the thesis. More specifically, the

attacks investigated in the thesis are diversified to profile variant aspects. More specifically, the proposed attacks are designed to be:

(1) **generalized:** every attack is conducted using various attacker's profiles to have a full view of the attack's impact on the designated overlay.

(2) **internally conducted:** i.e., the attack is launched internally from the overlay to cause a maximum damage to the overlay rather than externally shutting down other peers in the overlay.

(3) **localized:** a specific set of peers (victims) is *targeted*, i.e., the attacker's malicious resources are only directed towards attacking selective peers.

(4) **unlocalized:** complementing localized, in this scenario the attacker does not target a specific peer, i.e., the target of the attacker is to generally propagate disruptions to maximize the attacks impact on the overlay.

In the following sections the application of the proposed attacks with respect to the different P2P overlays is explained

7.1.1 Structured Overlays

For structured overlays, two main attack types were assessed, referred to as Localized Attacks (LA) and Routing Table Poisoning Attacks (RTP). We now discuss the main aspects concluded in the work presented in the thesis.

7.1.1.1 Localized Attacks (LA)

Structured overlays are mainly characterized by the closeness feature, where peers can relate to the locations of other peers, which in turn enhances the lookup forwarding procedure. However, progressive forms of localized attacks are shown to successfully exploit this feature even when divergent lookup mechanisms are in-operation, as we evaluated in Section 2.7.

Figure 7.1: Structured overlays: LA

As depicted in Figure 7.1, after assessing the impact of LAs, firstly, we designed a dynamic-based-voting detection mechanism to enhance divergent lookups which yielded successful detection rates close to 100%. Afterwards, a distributive eviction mechanism was proposed to restore the overlay's reliability, with a key feature of accurately propagating the information about malicious peers to the rest of the overlay. The proposed eviction mechanism was capable of accurately evict up to 90% of the malicious peers from the overlay while the attack is being conducted.

7.1.1.2 Routing Table Poisoning (RTP)

The second form of attacks conducted on structured overlays was the RTP attacks. RTP attacks serve as an inherent part of various existing attacks on structured overlays and consequently have a higher significance. In fact, the attacks main target is to contaminate honest peers routing tables with malicious peers propagating fake information. Notably, throughout the work presented in Section 4.6, we emphasized the severity of RTP attacks on structured overlays,

which reflects the likelihood of experiencing a similar drastic impact from related attacks such as sybil, flooding or eclipse attacks.

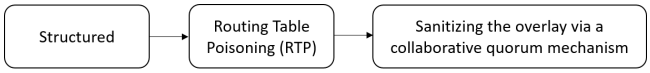


Figure 7.2: Structured overlays: RTP

Contemporary, we designed a distributive eviction mechanism where peers form a quorum once the detection mechanism, presented in Chapter 2, suspects a peer. Accordingly, peers are allowed to reach a collaborative decision on the suspected peer, which remarkably allows for a rapid sanitizing of the overlay via fully-partitioning of malicious peers from the rest of the overlay with success rates of up to 90%.

7.1.2 Unstructured Overlays

Unstructured overlays are inherently easy to maintain given the less constraints available when peers join or leave the overlay, i.e., peers are randomly assigned IDs which denotes that, unlike structured overlays, there are no notions of closeness or calculations required to determine next hops for forwarding a message from the sender to the receiver. In fact, the exhaustive searching is performed to *determine* the next hop leading to the destination peer. Hence, for large-scale applications, maintenance becomes more complex and resources/time consuming. To alleviate this, super peers are usually coupled with the unstructured overlays to form a super-unstructured overlay that allows for better organization of the messages forwarding procedure.

While integrating super peers is effective and efficient, however, this incentives attackers to attack super peers due to their higher impact on system. Therefore, we proposed an intuitive form of eclipse attacks, namely *Outgoing Eclipse Attacks (OEA)*, where malicious peers aim at delaying, forging or dropping outgoing messages from super peers. The attack were conducted using a localized and an unlocalized attack instances, which showed a significant impact as successfully delivered messages decreased in certain scenarios to around 5%.

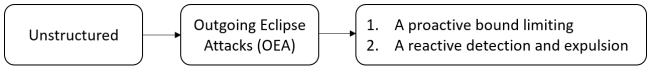


Figure 7.3: Unstructured overlays: OEA

On the other hand, to limit and counter OEAs, we designed a composite mechanism that pro-actively overcomes routing table infiltration through a monitoring technique that bounds the number of connections per peer. Simultaneously, a complementary reactive mechanism was proposed that distributively expels malicious peers from the honest peers routing tables with an accuracy rate of up to 99%.

7.1.3 Streaming Overlays

P2P streaming overlays fundamentally rely on the fact that peers participate in data dissemination process. This feature notably became a critical characteris-

tic for a wide area of streaming applications due to the reduced costs derived by the robust dissemination techniques. Consequently, attackers are continuously trying to exploit this feature via manipulating buffer-maps BM to inform neighboring peers about the (existing versus desired) data chunks.

In that context, in Chapter 6, we presented a severe, yet feasible, form of cheating attacks we refer to as *drop-chunk attack*. In this attack, malicious peers stop informing peers about the recent available chunks in their own buffer.

More importantly, as evaluated and proved in [NRS+16] peers are capable of inferring the overlay's topology. Adversely, we show that malicious peers can successfully place themselves in the closest positions to the stream source to receive chunks earlier than the rest of the overlay and thus, restrict further propagation of the chunks which in turn leads to a drastic degradation in the streaming quality. We showed that conducting such a composite internal-drop attack yielded a significant impact on the overlay where peers satisfaction level dropped to around 100% using a small fraction of malicious peers.

Figure 7.4: Streaming overlays: BM data dissemination cheating attacks

To restore peers satisfaction from the overlay, in a short time interval, we proposed in Section 6.3 a collaborative complaint system where neighboring peers agree on filing a complaint to the source when the average satisfaction level drops to a lower value than a predefined application specific threshold. Subsequently, the source validates the complaint and accordingly removes suspicious headnodes connections. Through a theoretical and a simulation study, we showed that our approach managed to effectively restore peers satisfaction up to 80-90% in severe attack conditions.

7.2 Summary: Research Questions & Contributions

Breaking down the work done in this thesis into three main research questions, with each research question addressing a certain P2P overlay, we show how attacks develop and evolve to significantly degrade the service provisioning. Accordingly, through the four contributions presented in the thesis, we designed and evaluated varied mitigation and detection techniques. In more details, we summarize and conclude the results of each research question and the corresponding contribution(s) in the following section.

Research Question (RQ1): How resilient are structured P2P overlays to attacks?

In this work, we address the severity of Localized EAs (LEAs) on structured P2P overlays. Through the rigorous evaluation detailed in Section 2.7, we highlighted that the decentralized design of P2P networks also exposes it to a variety of distributed threats with Eclipse Attacks (EAs) being a prominent attack impacting P2P functionalities.

On the other hand, although the basic technique of divergent lookups has been demonstrated as a countermeasure to mitigate EA, it can only (effectively) address limited variants of EAs, i.e., topology aware LEA (taLEA). As depicted in the results, the Lookup Success Ratio (LSR) drops down to 63% when malicious peers ratio is 25% of the network size. For more progressive and generic Localized Attack (LA) behaviors, as we showed in [IGS16], the LSR dramatically decreased to negligible values of around 1%, which remarkably highlights the severity of localized attacks on structured overlays.

Furthermore, in [IGS17], we emphasized that while distributiveness and scalability are attractive features, these facets also increase exposure to malicious peers which can propagate malicious routing information in structured overlays. Accordingly, a diverse set of continuously evolving attacks can be mounted that can cause severe service impairments over the entire overlay network.

We discussed in the related work in Section 4.2 that existing countermeasures focus on providing diversity or redundancy to overcome malicious routing information with their emphasis on periodic detection/removal mechanisms done locally within a peer as continuous monitoring or global sharing of peer status entails high costs. However, a local approach naturally also limits the global effectiveness prompting the need for distributed solutions. To tackle the aforementioned challenges, we presented two contributions, to counter LA and RTP attacks, respectively. First, to address LAs, we proposed a two-fold contribution as follows:

Contribution (C1):

***(C1.1)* Developing a highly accurate mechanism to detect malicious peers targeting a specific peer to attack.**

***(C1.2)* Developing an eviction mechanism that is generally applicable to various LA types while effectively evict malicious peers from the overlay.**

Next, we conclude the work done towards each of the aforementioned sub-contributions.

Contribution C1.1: In this first sub-contribution, we investigated both the detection and mitigation potential of enhanced divergent lookups for handling complex EA scenarios. In addition, we proposed an approach that can identify malicious peers with a high degree of accuracy.

In a nutshell, our simulations have shown EA mitigation rates of up to 96% in case 25% of the peers are malicious. Also, our approach allows for anonymity-fostering, fully decentralized usage, and facilitating downstream mechanisms such as malicious peer removal.

Contribution C1.2: The necessity of designing an eviction mechanism stems from the fact that:

> despite using a relatively small amount of attacking malicious peers, LAs severely impair the overall network's reliability due to the continuous evolving and mutating of the adversarial behaviors conducted by malicious peers.

Hence, as a countermeasure, we proposed a new two-fold LA countermeasure to evict malicious peers based on the detection mechanism proposed in [IGS15].

Our countermeasure has been evaluated in a comprehensive simulation experiment study using a generic LA model that covers a wide range of known LAs such as sybil, eclipse or poisoning attacks. The study showed reliability improvements of up to 97% in the presence of LAs, as well as successful evictions for up to 99% of the cases.

Next, In the context of RTP attacks, after emphasizing its impact on structured overlays, we successfully developed a sanitizing light-weight distributive sanitizing mechanism, as presented in the following contribution.

Contribution (C2): Sanitizing benign peers RT from malicious peers conducting RTP attack

To design a countermeasure against RTP attacks, we built upon contemporary distributed solutions (that developed specific attack detection and mitigation techniques for specific overlay types and specific attacks), to propose a generalized attack detection and mitigation approach applicable to varied overlay and attack models.

To that end, we proposed in [IGS17] a novel and efficient routing table sanitizing approach that (a) is independent of a specific attack variant, lookup approach or a specific victim set, (b) continuously detects and subsequently removes malicious routing information based on distributed quorum decisions, and (c) efficiently forwards malicious information findings to other peers which allows for progressive global sanitizing. The generalized mechanism showed a high sanitizing accuracy of up to 90% when evaluated against a generalized attack scenario with various adversarial behaviors.

Research Question (RQ2): How to defend against Outgoing Eclipse Attacks (OEAs) on unstructured super-P2P systems?

In this work, which is published in TrustCom 2018 [IRS18a], we highlighted the critical role of 'super peers' in unstructured overlays specially in large-scale applications which host millions of users. Given the criticality and impact of super peers, attackers in turn targeted super peers due to the resultant significant damage on the corresponding service offered through the unstructured overlay based application.

We considered the prominent class of Outgoing Eclipse Attacks (OEAs) where an attacker aims at blocking the communication by controlling all the outgoing connections of honest super peers. Our interest on OEA stems from the fact that our simulation studies reveal that OEAs can cause up to 90% of all service requests to fail.

Contribution (C3): An effective detection and peer eviction mechanism to mitigate OEAs in super-P2P systems.

In order to diminish the severity of OEAs on super peers, we proposed a detection mechanism that relies upon a novel (a) monitoring and (b) malicious peer

eviction scheme based on a composite proactive and reactive mechanism. Our proactive mechanism enforced an upper bound on the number of connections an attacker can establish, whereas our reactive mechanism expelled malicious peers from the overlay using a distributed consensus protocol.

We showed that our protection mechanism is highly effective and exhibits a low false-positive rate. Our extensive simulation study validated the analytical results over a large range of parameters with observed detection accuracies of 99% and throughput enhancements of up to 100% while entailing an overhead of less than 5%.

Research Question (RQ3): For on-line data streaming overlays, what is the impact of internal Denial-of-Service (DoS) attacks on the streaming quality and how to mitigate cheating attacks in such a tightly QoS constrained environment?

In this research question, we investigated the resiliency of online streaming P2P overlays, which provide a popular service for data-intensive applications such as video streaming. First, we showed that, although the attractive features acquired when involving users with data dissemination, they entailed security risks including a variety of denial-of-service attacks. In other words, we demonstrated (a) the feasibility of conducting such attacks, and (b) that while extensive research exists on mitigating varied attack types, their effectiveness is limited if the attacker can infer information about the topology such as the identity of nodes that have direct connections to the source.

Through simulations, we illustrated how the attacker can leverage the gained insights, such as the inference techniques described in [NRS+16], to place malicious participants in prominent positions. Consequently, by dropping chunks that should be forwarded, the malicious peers degraded the performance in a stealthy way that does not raise suspicion causing average chunk loss rates up to 100%. To that end, we proposed a detection technique aiming at restoring the users satisfaction levels from the streaming service, as discussed in the last contribution of the thesis, as concluded below.

Contribution (C4): Designing a detection mechanism to counter the impact of data dissemination (BM) cheating attacks.

Tackling research question 3, we proposed in [IRS18b] a detection mechanism that identifies DoS cheating attacks attacks, more specifically to online streaming overlays is the BM cheating attack. Thus, the detection mechanism's role was to accurately remove potential malicious peers from their disruptive positions.

We ascertained, theoretically and through simulations, that malicious peers cannot misuse the detection mechanism to gain influence. Our simulation-based study indicated that the proposed detection mechanism is able to detect malicious peers with up to 80-90% accuracy while inducing a small overhead of approximately 8%.

Bibliography

[AAM+15] M. Amad, D. Aıssani, A. Meddahi, N. Madi, and R. Bouiche. "A Priority Based Lookup Model for VoIP Applications in Unstructured P2P Networks". In: *Proceedings of IPAC*. 2015, 35:1–35:5.

[AEO12] F. de Asís López-Fuentes, I. Eugui-De-Alba, and O. M. Ortíz-Ruiz. "Evaluating P2P Networks against Eclipse Attacks". In: *Proceedia Technology* 3 (2012), pp. 61–68.

[APR+15] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. "Distributed Agreement in Dynamic Peer-to-Peer Networks". In: *Journal of Computer and System Sciences* 81.7 (2015), pp. 1088–1109.

[Ber06] D. J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *Proceedings of Public Key Cryptography (PKC)*. 2006, pp. 207–228.

[BG03] B. Beverly Yang and H. Garcia-Molina. "Designing a Super-peer Network". In: *Proceedings of International Conference on Data Engineering*. 2003, pp. 49–60.

[BHK07] I. Baumgart, B. Heep, and S. Krause. "OverSim: A Flexible Overlay Network Simulation Framework". In: *Proceedings of INFOCOM*. 2007, pp. 79–84.

[BJ91] F. Belli and P. Jedrzejowicz. "Comparative Analysis of Concurrent Fault-Tolerance Techniques for Real-Time Applications". In: *Proceedings of ISSRE*. 1991, pp. 202–209.

[BM07] I. Baumgart and S. Mies. "S/Kademlia: A Practicable Approach Towards Secure Key Based Routing". In: *Proceedings of ICPADS* (2007), pp. 1–8.

[BS06] S. Baset and H. Schulzrinne. "An Analysis of The Skype P2P Internet Telephony Protocol". In: *Proceedings of INFOCOM*. 2006, pp. 1–11.

[CCF+12] T. Cholez, I. Chrisment, O. Festor, and G. Doyen. "Detection and Mitigation of Localized Attacks in a Widely Deployed P2P Network". In: *Peer-to-Peer Networking and Applications* 6.2 (2012), pp. 155–174.

[CCF10] T. Cholez, I. Chrisment, and O. Festor. "Efficient DHT attack mitigation through peers' ID distribution". In: *Proceedings of IPDPSW* (2010), pp. 1–8.

[CDG+02] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. "Secure Routing for Structured Peer-to-peer Overlay Networks". In: *SIGOPS Oper. Syst. Rev.* 36.SI (2002), pp. 299–314.

[CL02] M. Castro and B. Liskov. "Practical Byzantine Fault Tolerance and Proactive Recovery". In: *In ACM Transactions on Computer Systems (TOCS)* 20.4 (2002), pp. 398–461.

[CL99] M. Castro and B. Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (1999), pp. 173–186.

[CLW07] Y. Cui, D. Li, and J. Wu. "Impact of Buffer Map Cheating on the Streaming Quality in DONet". In: *Proceedings of ICCS*. 2007, pp. 817–824.

[CML+06] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira. "HQ Replication: A Hybrid Quorum Protocol for Byzantine Fault Tolerance". In: *Proceedings of Operating Systems Design and Implementation*. 2006, pp. 177–190.

[CRB+03] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. "Making Gnutella-like P2P Systems Scalable". In: *Proceedings of SIGCOMM*. 2003, pp. 407–418.

[DH06] J. Dinger and H. Hartenstein. "Defending The Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration". In: *Proceedings of ARES*. 2006, 8–pp.

[DIV18] K. Demir, H. Ismail, and T. Vateva-Gurova. "Securing the Cloud Assisted Smart Grid". In: *Internal Journal of Critical Infrastructure Protection* (2018).

[DKC08] T. Dimitriou, G. Karame, and I. Christou. "SuperTrust-A Secure and Efficient Framework for Handling Trust in Super Peer Networks". In: *Proceedings of ICDCN*. 2008, pp. 350–362.

[DKS12] R. L. D. Germanus, A. Khelil, and N. Suri. "Susceptibility Analysis of Structured P2P Systems to Localized Eclipse Attacks". In: *Proceedings of SRDS* (2012), pp. 11–20.

[DM09] G. Danezis and P. Mittal. "SybilInfer: Detecting Sybil Nodes using Social Networks". In: *Proceedings of NDSS* (2009), pp. 1–15.

[Dou02] J. R. Douceur. "The Sybil Attack". In: *Proceedings of the international workshop on peer-to-peer systems*. 2002, pp. 251–260.

[EMG14] F. A. Eichert, M. Monhof, and K. Graffi. "The Impact of Routing Attacks on Pastry-Based P2P Online Social Networks". In: *Proceedings of Euro-Par*. 2014, pp. 347–358.

[FGG15] A. Fiandrotti, R. Gaeta, and M. Grangetto. "Simple countermeasures to mitigate the effect of pollution attack in network coding-based peer-to-peer live streaming". In: *IEEE Transactions on Multimedia* 17 (2015), pp. 562–573.

[Fis83] M. J. Fischer. "The Consensus Problem in Unreliable Distributed Systems". In: *Proceedings of International Conference on Fundamentals of Computation Theory*. Lecture Notes in Computer Science (1983), pp. 127–140.

[FMR+09] R. Fantacci, L. Maccari, M. Rosi, L. Chisci, L. M. Aiello, and M. Milanesio. "Avoiding Eclipse Attacks on Kad/Kademlia: an Identity Based Approach". In: *Proceedings of ICC*. 2009, pp. 1–5.

[FTT10] L. Fan, P. Trinder, and H. Taylor. "Design Issues for Peer-to-Peer Massively Multiplayer Online Games". In: *Proceedings of IJAMC* 4.2 (2010), pp. 108–125.

[GD16] I. Gkortsilas and K. Deltouzos. "Detecting and Isolating Pollution Attacks in Peer-to-peer VoD Systems". In: *Proceedings of EuCNC*. 2016, pp. 340–344.

[GE12] J. S. Gilmore and H. A. Engelbrecht. "A Survey of State Persistency in Peer-to-Peer Massively Multiplayer Online Games". In: *IEEE Transactions on Parallel and Distributed Systems* 23.5 (2012), pp. 818–834.

[GG13] R. Gaeta and M. Grangetto. "Identification of Malicious Nodes in Peer-to-Peer Streaming: A Belief Propagation-Based Technique". In: *IEEE Transactions on Parallel & Distributed Systems* 10 (2013), pp. 1994–2003.

[GIS15] D. Germanus, H. Ismail, and N. Suri. "PASS: An Address Space Slicing Framework for P2P Eclipse Attack Mitigation". In: *IEEE Symposium on Reliable Distributed Systems (SRDS)*. 2015, pp. 74–83.

[GK03] N. S. Good and A. Krekelberg. "Usability and Privacy: A Study of Kazaa P2P File-sharing". In: *Proceedings of SIGCHI Conference on Human Factors in Computing Systems* (2003), pp. 137–144.

[GK14] S. Gite and P. Kamat. "Critical Analysis of P2P Communication, Security Concerns and Solutions". In: (2014), 30:899–30:909.

[GRS+14] D. Germanus, S. Roos, T. Strufe, and N. Suri. "Mitigating Eclipse Attacks in Peer-to-Peer Networks". In: *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*. 2014.

[GSC91] R. E. Gantenbein, S. Y. Shin, and J. R. Cowles. "Evaluation of Combined Approaches to Distributed Software Based Fault Tolerance". In: *Proceedings of PRDC* (1991), pp. 70–75.

[HC04] A. Habib and J. Chuang. "Incentive Mechanism for Peer-to-peer Media Streaming". In: *Proceedings of IWQOS*. 2004, pp. 171–180.

[HC15] H. Hsieh and M. Chiang. "New Approach to Improve the Generalized Byzantine Agreement Problem". In: *In International Journal of Computer Theory and Engineering* 7.2 (2015), p. 120.

[HFC+08] Y. Huang, T. Z. Fu, D.-M. Chiu, J. Lui, and C. Huang. "Challenges, Design and Analysis of a Large-scale P2P-vod System". In: *Proceedings of ACM SIGCOMM*. 2008, pp. 375–388.

[IGS15] H. Ismail, D. Germanus, and N. Suri. "Detecting and Mitigating P2P Eclipse Attacks". In: *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*. 2015, pp. 224–231.

[IGS16] H. Ismail, D. Germanus, and N. Suri. "Malicious Peers Eviction for P2P Overlays". In: *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*. 2016, pp. 216–224.

[IGS17] H. Ismail, D. Germanus, and N. Suri. "P2P Routing Table Poisoning: A Quorum-based Sanitizing Approach". In: *Computers & Security* 65 (2017), pp. 283–299.

[IRS18a] H. Ismail, S. Roos, and N. Suri. "A Composite Malicious Peer Eviction Mechanism for Super-P2P Systems". In: *IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom)*. 2018.

[IRS18b] H. Ismail, S. Roos, and N. Suri. "A Detection Mechanism for Internal Attacks on Pull-based P2P Streaming Systems". In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2018.

[KAL+15] J. Kos, M. Aiash, J. Loo, and D. Trček. "U-Sphere: Strengthening scalable flat-name routing for decentralized networks". In: *Computer Networks* 89 (2015), pp. 14–31.

[KBC+00] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. "OceanStore: An Architecture for Global-scale Persistent Storage". In: *ACM SIGPLAN Not.* 35.11 (2000), pp. 190–201.

[KCW10] J. Kong, W. Cai, and L. Wang. "The Evaluation of Index Poisoning in Bittorrent". In: *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on*. 2010, pp. 382–386.

[KLK+12] H. Koo, Y. Lee, K. Kim, B.-h. Roh, and C. Lee. "A DDoS Attack by Flooding Normal Control Messages in Kad P2P Networks". In: *Proceedings of the International Conference on Advanced Communication Technology (ICACT)* (2012), pp. 213–216.

[KLR09] M. Kohnen, M. Leske, and E. P. Rathgeb. "Conducting and Optimizing Eclipse Attacks in the Kad Peer-to-Peer Network". In: *International Conference on Research in Networking*. 2009, pp. 104–116.

[KT08] A. Kapadia and N. Triandopoulos. "Halo: High-Assurance Locate for Distributed Hash Tables". In: *Proceedings of The Network and Distributed System Symposium (NDSS)* (2008), p. 142.

[KW14] X. Kang and Y. Wu. "A Trust-based Pollution Attack Prevention Scheme in Peer-to-peer Streaming Networks". In: *Computer Networks* 72 (2014), pp. 62–73.

[LC08] Z. Li and X. Chen. "Misusing Kademlia Protocol to Perform DDoS Attacks". In: *Proceedings of International Symposium on Parallel and Distributed Processing with Applications (ISPA)* (2008), pp. 80–86.

[LCP+05] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes". In: *IEEE Communications Surveys Tutorials* 7.2 (2005), pp. 72–93.

[LGL08] Y. Liu, Y. Guo, and C. Liang. "A Survey on Peer to Peer Video Streaming Systems". In: *Peer-to-peer Networking and Applications* 1.1 (2008), pp. 18–28.

[LKC+12] Y. Lee, H. Koo, S. Choi, B.-h. Roh, and C. Lee. "Advanced Node Insertion Attack with Availability Falsification in Kademlia-based P2P Networks". In: *Proceedings of ICACT*. 2012, pp. 73–76.

[LL10] Y. Li and J. C. Lui. "Stochastic Analysis of A Randomized Detection Algorithm for Pollution Attack in P2P Live Streaming Systems". In: *Performance Evaluation* 67.11 (2010), pp. 1273–1288.

[LLR09] C. Liang, Y. Liu, and K. W. Ross. "Topology Optimization in Multi-tree based P2P Streaming System". In: *Proceedings of ICTAI*. 2009, pp. 806–813.

[LMG+10] H. Lin, R. Ma, L. Guo, P. Zhang, and X. Chen. "Conducting Routing Table Poisoning Attack in DHT Networks". In: *Proceedings of ICCCAS*. 2010, pp. 254–258.

[LML13] C. Lu, X. Miao, and Z. Liu. "A Safety Algorithm of P2P Routing based on Multiple-Encryption Detecting Technology". In: *Indonesian Journal of Electrical Engineering and Computer Science* 11.10 (2013), pp. 5815–5823.

[LMS+10] T. Locher, D. Mysicka, S. Schmid, and R. Wattenhofer. "Poisoning the Kad network". In: *Proceedings of the International Conference on Distributed Computing and Networking* 5935 (2010), pp. 195–206.

[LNR06] J. Liang, N. Naoumov, and K. W. Ross. "The Index Poisoning Attack in P2P File Sharing Systems". In: *Proceedings of INFOCOM*. 2006, pp. 1–12.

[LWC09] D. Li, J. Wu, and Y. Cui. "Defending Against Buffer Map Cheating in DONet-Like P2P Streaming". In: *IEEE Trans. Multimedia*. 2009, pp. 535–542.

[LYL14] Q. Li, J. Yu, and Z. Li. "An Enhanced Kad Protocol Resistant to Eclipse Attacks". In: *Proceedings of the International Conference on Networking, Architecture, and Storage (NAS)* (2014), pp. 83–87.

[LZL+05] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, and J. Li. "Scalable Supernode Selection in Peer-to-Peer Overlay Networks". In: *Proceedings of Hot Topics in Peer-to-Peer Systems (Hot-P2P)*. 2005, pp. 18–25.

[MB09] P. Mittal and N. Borisov. "Shadowwalker: Peer-to-Peer Anonymous Communication Using Redundant Structured Topologies". In: *Proceedings of ACM CCS*. 2009, pp. 161–172.

[MB12] P. Mittal and N. Borisov. “Information Leaks in Structured Peer-to-Peer Anonymous Communication Systems”. In: *Transactions on Information and System Security (TISSEC)* 15.1 (2012), p. 5.

[MHC06] S. Min, J. Holliday, and D. Cho. “Optimal Super-peer Selection for Large-scale P2P System”. In: *Proceedings of ICHI)*. 2006, pp. 588–593.

[MM02] P. Maymounkov and D. Mazieres. “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric”. In: *Proceedings of IPTPS*. 2002, pp. 53–65.

[MTH+09] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. “Scalable Onion Routing with Torsk”. In: *Proceedings of ACM CCS*. 2009, pp. 590–599.

[NF15] T. S. Nguyen Giang Stefanie Roos and M. Fischer. “RBCS: A resilient Backbone Construction Scheme for Hybrid Peer-To-Peer Streaming”. In: *Proceedings of LCN*. 2015, pp. 261–269.

[NFS13] G. Nguyen, M. Fischer, and T. Strufe. “Ossim: A Generic Simulation Framework for Overlay Streaming”. In: *Proceedings of SCSC*. 2013, 30:1–30:8.

[NFS14] G. Nguyen, M. Fischer, and T. Strufe. “On the Resilience of Pull-based P2P Streaming Systems Against DoS Attacks”. In: *Proceedings of Stabilization, Safety, and Security of Distributed Systems (SSS)*. 2014, pp. 33–47.

[NR06] N. Naoumov and K. Ross. “Exploiting P2P Systems for DDoS Attacks”. In: *Proceedings of International Conference on Scalable Information Systems* (2006), p. 47.

[NRS+16] G. Nguyen, S. Roos, B. Schiller, and T. Strufe. “SWAP: Protecting Pull-based P2P Video Streaming Systems from Inference Attacks”. In: *Proceedings of WoWMoM*. 2016, pp. 1–9.

[NW06] A. Nambiar and M. Wright. “Salsa: A structured Approach to Large-scale Anonymity”. In: *Proceedings of ACM CCS* (2006), pp. 17–26.

[OC01] E. Oh and J. Chen. “Parallel Routing in Hypercube Networks with Faulty Nodes”. In: *Proceedings of ICPADS*. 2001, pp. 338–345.

[Pon93] G. Pongor. “OMNeT: Objective Modular Network Testbed”. In: *Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems (MASCOTS)*. 1993, pp. 323–326.

[PRR09] A. Panchenko, S. Richter, and A. Rache. “NISAN: Network Information Service for Anonymization Networks”. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. 2009, pp. 141–150.

[RL03] R. Rodrigues and B. Liskov. “Rosebud: A Scalable Byzantine Fault Tolerant Storage Architecture”. In: *MIT LCS TR/932* (2003).

[RPI12] N. Ramzan, H. Park, and E. Izquierdo. “Video Streaming over P2P Networks: Challenges and Opportunities”. In: *Signal Processing: Image Communication* 27 (2012), pp. 401–411.

[RSV+15] C. Rottondi, M. Savi, G. Verticale, and C. Krauß. "Mitigation of Peer-to-peer Overlay Attacks in the Automatic Metering Infrastructure of Smart Grids". In: *Security and Communication Networks* 8.3 (2015), pp. 343–359.

[SCD+04] A. Singh, M. Castro, P. Druschel, and A. Rowstron. "Defending against Eclipse Attacks on Overlay Networks". In: *Proceedings of SIGOPS*. 2004, pp. 115–120.

[Sch05] C. Scheideler. "How to Spread Adversarial Nodes?: Rotate!" In: *Proceedings of ACM STOC*. 2005, pp. 704–713.

[SEB07] M. Steiner, T. En-Najjary, and E. W. Biersack. "Exploiting KAD: Possible Uses and Misuses". In: *ACM SIGCOMM Computer Communication Review* 37.5 (2007), pp. 65–70.

[SF12] S. Sen and M. Freedman. "Commensal Cuckoo: Secure Group Partitioning for Large-scale Services". In: *In Operating Systems Review (OSR) in ACM Special Interest Group on Operating Systems (SIGOPS)* 46.1 (2012), pp. 33–39.

[SMK+01] 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 the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '01. 2001, pp. 149–160.

[SND+06] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. "Eclipse Attacks on Overlay Networks: Threats and Defenses". In: *Proceedings of International Conference on Computer Communications (INFOCOM)*. 2006, pp. 1–12.

[SR12] J. K. So and D. S. Reeves. "Antiliar: Defending Against Cheating Attacks in Mesh Based Streaming". In: *Proceedings of P2P*. 2012, pp. 115–125.

[SS09] C. Scheideler and S. Schmid. "A Distributed and Oblivious Heap". In: *Journal of International Colloquium on Automata, Languages, and Programming (ICALP)*. 2009, pp. 571–582.

[ST15] D. Sharma and S. Thakur. "Performance Analysis of Sybil Decline: Attack Detection and Removal Mechanism in Social Network". In: *Proceedings of IJCSIS* 13 (2015), p. 165.

[STR10a] X. Sun, R. Torres, and S. Rao. "Preventing DDoS Attacks on Internet Servers Exploiting P2P Systems". In: *Computer Networks* 54.15 (2010), pp. 2756–2774.

[STR10b] X. Sun, R. Torres, and S. G. Rao. "On the Feasibility of Exploiting P2P Systems to Launch DDoS Attacks". In: *Peer-to-Peer Networking and Applications* 3 (2010), pp. 36–51.

[Tha13] S. M. Thampi. "A Review on P2P Video Streaming". In: *CoRR* abs/1304.1235 (2013).

[TLH14] H.-Y. Teng, C.-N. Lin, and R.-H. Hwang. "A Self-similar Superpeer Overlay Construction Scheme for Super Large-scale P2P Applications". In: vol. 16. 1. 2014, pp. 45–58.

[UPS11] G. Urdaneta, G. Pierre, and M. V. Steen. "A Survey of DHT Security Techniques". In: *In ACM Computing Surveys (CSUR)* 43.2 (2011), pp. 1–53.

[VAM+06] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. "A Hierarchical Characterization of a Live Streaming Media Workload". In: *ACM Transactions on Networking* 14 (2006), pp. 133–146.

[VGL+07] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. "Measurement and Modeling of a Large-scale Overlay for Multimedia Streaming". In: *Proceedings of QSHINE*. 2007, 3:1–3:7.

[VGV05] S. Voulgaris, D. Gavidia, and M. Van Steen. "Cyclon: Inexpensive Membership Management for Unstructured P2P Overlays". In: *Journal of Network and Systems Management* 13.2 (2005), pp. 197–217.

[WK13] L. Wang and J. Kangasharju. "Measuring Large-scale Distributed Systems: Case of Bittorrent Mainline DHT". In: *IEEE P2P 2013 Proceedings*. 2013, pp. 1–10.

[WTC+08] P. Wang, J. Tyra, E. Chan-Tin, T. Malchow, D. F. Kune, N. Hopper, and Y. Kim. "Attacking the Kad Network". In: *Proceedings of SecureComm*. 2008, pp. 1–10.

[YK13] A. Yahyavi and B. Kemme. "Peer-to-Peer Architectures for Massively Multiplayer Online Games: A Survey". In: *ACM Computing Surveys (CSUR)* 46.1 (2013), 9:1–9:51.

[YKG+10] M. Young, A. Kate, I. Goldberg, and M. Karsten. "Practical Robust Communication in DHTs Tolerating a Byzantine Adversary". In: *2010 IEEE 30th International Conference on Distributed Computing Systems*. 2010, pp. 263–272.

[ZCB96] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. "How to Model an Internetwork". In: *Proceedings of INFOCOM*. 1996, pp. 594–602.

[ZJT13] S. Zargar, J. Joshi, and D. Tipper. "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks". In: *IEEE Communications Surveys & Tutorials* 15 (2013), pp. 2046–2069.

[ZL06] W. X. Z. Yao and D. Loguinov. "Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks". In: *Proceedings of ICNP*. 2006, pp. 32–41.

[ZLL+05] X. Zhang, J. Liu, B. Li, and Y.-S. Yum. "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-peer Live Media Streaming". In: *Proceedings of INFOCOM*. 2005, pp. 2102–2111.

[ZSD12] G. Zhan, W. Shi, and J. Deng. "Design and Implementation of TARF: A trust-aware Routing Framework for WSNs". In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 9 (2012), pp. 184–197.

[ZXW+14] J. Zhang, W. Xing, Y. Wang, and D. Lu. “Modeling and Performance Analysis of Pull-based Live Streaming Schemes in Peer-to-Peer Network”. In: *Computer Communications* 40 (2014), pp. 22–32.