SECURITY AND ECONOMIC IMPLICATIONS OF LOCALIZING TRAFFIC IN OVERLAY NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Jeffrey C. Seibert

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2012

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

ABSTRACT viii

1 INTRODUCTION 1
- 1.1 Motivation 2
- 1.2 Thesis Focus and Contributions 3
- 1.3 Thesis Roadmap 5

2 SYSTEM AND ATTACKER MODEL 6
- 2.1 Internet Model 6
- 2.2 Virtual Coordinate System Model 7
- 2.3 Mesh Overlay Model 7
- 2.4 Attacker Model 9

3 THE INTERNET-WIDE IMPACT OF P2P TRAFFIC LOCALIZATION ON ISP PROFITABILITY 10
- 3.1 Introduction 10
- 3.2 Modeling Inter-AS P2P Traffic 14
  - 3.2.1 The Gravity Model 14
  - 3.2.2 The Affinity Model 15
  - 3.2.3 Model Validation 16
- 3.3 Modeling P2P Localization 21
  - 3.3.1 System Architecture Assumptions 22
  - 3.3.2 Locality Models 23
- 3.4 Measuring ISP Profitability 25
  - 3.4.1 An Ideal Metric for ISP Profitability 26
  - 3.4.2 Approximating ISP Profitability 27
  - 3.4.3 Pricing Models 29
- 3.5 Impact of Localization Policies 33
  - 3.5.1 Profitability before Localization 34
  - 3.5.2 Localization Deployed by Individual ASes 37
  - 3.5.3 Internet-wide Localization Deployed 40
  - 3.5.4 Localization Deployed by Large ASes 42
- 3.6 Impact of Business-Relationship Policies 44
  - 3.6.1 Modeling Business-Relationship Based Policies 44

Page
3.6.2 Best Strategy for Individual ASes 45
3.6.3 Internet Impact of Business-Relationship Based Policies . . . 47
3.7 Summary 48
4 SECURING VIRTUAL COORDINATES BY ENFORCING PHYSICAL LAWS 53
4.1 Introduction 53
4.2 Vivaldi Coordinate System 56
4.3 Attacks Against VCS 57
4.4 Description of Newton 58
4.4.1 Vivaldi as a Physical System 58
4.4.2 Using Physical Laws to Identify Invariants 59
4.4.3 Detecting Extraneous Indirect Forces 60
4.4.4 Detecting Extraneous Direct Forces 64
4.4.5 Using IN1, IN2, and IN3 to Design Newton 65
4.5 Simulation Results 69
4.5.1 Attacks Mitigation 71
4.5.2 Extreme Attack Scenarios 75
4.5.3 Newton in Higher-dimensional and Hyperbolic Space 78
4.5.4 Invariants under Attack 79
4.6 Experimental Results 80
4.6.1 Performance in Benign Networks 80
4.6.2 Attack Mitigation 81
4.7 Summary 82
5 A DESIGN FOR SECURING DATA DELIVERY IN MESH-BASED PEER-TO-PEER STREAMING 92
5.1 Introduction 92
5.2 Mesh-Based Peer-to-Peer Streaming 96
5.3 Attacks Against Data Delivery 97
5.3.1 Attack Model 97
5.3.2 Attacks on the Data Plane 98
5.3.3 Attacks on the Control Plane 100
5.3.4 Our Focus 101
5.4 A Design For Securing Data Delivery 101
5.4.1 Design Goals and Overview 102
5.4.2 Protecting Peers through Local Reputation 103
5.4.3 Source Protection with Health Monitoring 108
5.4.4 Rate-limiting Bootstrap 110
5.5 Security Analysis 111
5.5.1 Attacks on Data Score Calculation 111
5.5.2 Attacks on Graph Connectivity Score Calculation 113
5.5.3 Other Attacks 114

Page
5.6 Experimental Methodology 115
5.6.1 Schemes Considered 115
5.6.2 Attacks Considered 116
5.6.3 Experiment Configuration 117
5.6.4 Performance Metrics 119
5.7 Experimental Evaluation 119
5.7.1 Robust Neighbor Selection 119
5.7.2 Source Protection 122
5.7.3 Rate-limiting Bootstrap 123
5.7.4 Data Delaying 126
5.8 Summary 127
6 RELATED WORK 129
6.1 P2P Localization 129
6.2 Virtual Coordinate Systems 130
6.3 P2P Streaming 131
7 CONCLUSION 134
LIST OF REFERENCES 136
VITA 146

LIST OF TABLES

Table Page

3.1 Approximating the 95th percentile for incoming traffic 28

3.2 Pricing functions 30

3.3 ASes profiting or losing by category (no localization) 34

3.4 Interpreting metric results 35

4.1 Sensitivity on threshold for IN2 67

5.1 Attacks against data and control planes 99

5.2 Mechanisms for each component of system 115

5.3 Notation 118

LIST OF FIGURES

Figure Page

2.1 ASes and their different business models 6

2.2 Example of a unidirectional mesh-based streaming overlay in which the source sends two different data chunks as denoted by the gray triangle and orange square. Each node has an in-neighbor set and an out-neighbor set. For example, for node 6, the in-neighbor set consists of node 7 and the source, while the out-neighbor set consists of nodes 1 and 9. 8

3.1 Affinity and Gravity models compared against real measurements. . . . 19

3.2 Relative error for Affinity and Gravity models 20

3.3 Example topology illustrating our pricing model. P95 refer to P2P traffic. 30

3.4 Individual AS deploys localization with Single(history) locality model. Sensitivity to pricing models. 36

3.5 Individual AS deploys localization with Class pricing model. Sensitivity to locality models. 38

3.6 All ASes deploy localization with Class pricing model. Sensitivity to locality models. 49

3.7 Largest ASes deploy localization with Single(history) locality model. Sensitivity to pricing models. 50

3.8 Individual AS deploys Business, Single(history) or Hybrid, with Class pricing model. 51

3.9 All ASes deploy Business, Single(history), Hybrid or Individual Best, with Class pricing model. 52

4.1 Detecting extraneous indirect forces 61

4.2 Invariants shown through deployments of Vivaldi on PlanetLab. 66

4.3 Simulation results – inflation attack 69

4.4 Simulation results – deflation attack 70

4.5 Simulation results – oscillation attack 71

4.6 Simulation results – frog-boiling attack 72

Figure Page

4.7 Centroid over time for frog-boiling attack 74

4.8 Simulation results – network-partition attack 75

4.9 Simulation results – high percentage of attackers 76

4.10 Simulation results – attacks start at the beginning 77

4.11 Simulation results – accuracy when using 4 dimensions in hyperbolic space with 30% attackers 78

4.12 Simulation results – attackers (30%) push the limits of the thresholds used by the three invariants 84

4.13 Simulation results – attackers (30%) rotate around the origin at a slow rate 85

4.14 PlanetLab results – no attackers 86

4.15 PlanetLab results – accuracy of Newton for 4 days 86

4.16 PlanetLab results – inflation attack 87

4.17 PlanetLab results – deflation attack 88

4.18 PlanetLab results – oscillation attack 89

4.19 PlanetLab results – frog-boiling attack 90

4.20 PlanetLab results – network-partition attack 91

5.1 Importance of peer protection. 120

5.2 Importance of source and bootstrap protection. 121

5.3 Evaluating the corruption factor in different scenarios. 124

5.4 Peers running the local reputation scheme while attackers conduct data delaying attacks. 127

ABSTRACT

Seibert, Jeffrey C. Ph.D., Purdue University, May 2012. Security and Economic Implications of Localizing Traffic in Overlay Networks. Major Professor: Cristina Nita-Rotaru.

Overlay networks are a collection of nodes that form a virtual network on top of the normal routing infrastructure of the Internet. These virtual networks allow nodes to organize themselves for the purpose of transferring data in a robust manner. Overlay networks, and in particular Peer-to-Peer (P2P) systems, have become very popular as they provide scalable services for content distribution. However, many P2P systems have been oblivious to network locality, thus causing an increase in the amount of traffic that must leave an Internet Service Provider (ISP). P2P localization has then been proposed as a solution to contain traffic to within an ISP. In this dissertation, we first study the economic impact of actually deploying localization at an Internet-wide scale. We then consider how insider attackers can disrupt localization services and study how to protect such services from attacks. Finally, as insiders can also attack the overlays that utilize localization, we propose defenses for mitigating attacks in a high-bandwidth P2P streaming system.

1 INTRODUCTION

Distributed systems have become an integral part of many of the services that run on top of the Internet. Particularly, overlay networks, such as Peer-to-Peer (P2P) systems and Content Distribution Networks (CDN), represent a large fraction of traffic on the Internet today as they allow for scalable distribution of content. For example, BitTorrent [1] and YouTube [2], which represent a popular P2P and CDN system respectively, both have millions of users.

Overlay networks are a collection of nodes that form a virtual network on top of the normal routing infrastructure of the Internet. These virtual networks allow nodes to organize themselves for the purpose of transferring data in an efficient and robust manner. Overlay networks have been designed and deployed for two broad categories of purpose, routing and content distribution.

Routing around failures on the Internet was the first motivation to implement and use overlays [3]. However, overlays have also been used for routing in the context of voice over IP (e.g. Skype [4]) and also for providing anonymous communications using onion routing (e.g. Tor [5]). In general, these types of overlays provide different means of one-to-one communication.

Content distribution has also been a major motivator in the design of overlays. Content distribution comes mainly in two different forms: file distribution and streaming. The main difference between the two being that in file distribution, the entire file must first be downloaded before using, while streaming allows media such as music or video to be watched while downloading. Overlays used for content distribution can be divided into two categories. The first are P2P systems, where the users who are downloading the content also upload the content to others, expending their own resources to provide service to others. Many P2P systems have been proposed and deployed for file distribution, such as BitTorrent [1] and others [6] and also for streaming, such as PPLive [7] and others [8–15]. CDNs are also used for content distribution, these overlays are owned by a single entity who place servers all over the world. Then, when a user desires to download content, they will be directed to a server that is close to them. In CDNs, unlike P2P systems, the users do not upload data to others. CDNs are also used for both file distribution and streaming. For example, Akamai [16] is mainly used for file distribution and YouTube [2] also uses its own CDN to stream videos to users.

1.1 Motivation

While many overlay networks such as CDNs, have attempted to efficiently distribute content, many popular P2P systems that utilize a large amount of bandwidth have been oblivious to network locality. This lack of network-awareness has inevitably lead to an increase of traffic that leaves an Internet Service Provider (ISP). As many ISPs must pay others to carry traffic for them, the extra traffic increases their costs significantly. Due to this increase, many ISPs have tried to limit or block P2P traffic. In response, researchers have proposed several P2P localization schemes, with the goal of containing traffic to within an ISP. P2P localization has since become a critical component for designing distributed systems as it provides better performance by using network locality as an optimization criteria and allows them to better utilize network resources. As a result, localization can have a significant impact on the global Internet. However, as P2P localization is not widely deployed on the Internet today, it is still not well understood who will benefit from it and to what degree.

Furthermore, localization services may be subject to attack themselves, as insider attackers may try to disrupt them for either economic gains or malicious purposes. As localization services typically provide network-awareness, this could cause overlays to suffer in terms of performance, as overlays may not match the underlying network topology. Thus it is also important to ensure such services can defend against attacks.

Finally, adversaries may target the application-level overlays that utilize localization. By hijacking control of overlay traffic, attackers can cause problems such as denying service to users. Therefore, even though the overlay may be designed for high-bandwidth data delivery in benign scenarios, the application could suffer by receiving very little data due to malicious attackers. Creating robust overlays that can survive such threats thus is also an important problem.

1.2 Thesis Focus and Contributions

This thesis focuses on the design and implementation of secure and robust distributed systems that can continue to meet performance goals in the face of attacks. This work is motivated by the fact that even though distributed systems are ubiquitous and enable a plethora of applications that are currently in use today, they often can be manipulated by attackers, causing them to be practically unusable. The reality and threat of attackers has caused many to consider how to integrate security principles into the design of distributed systems, while still being able to meet the performance needs of a scalable system. This thesis centers around building intrusion-tolerant, low-latency, high-bandwidth distributed systems and understanding the security-related effects of distributed systems on the Internet ecosystem.

Specifically, my thesis work has focused on three directions: (1) **Internet-wide impact of localization:** as with any new disruptive technology, P2P localization can affect the Internet ecosystem in unforeseen ways, resulting in unintended side-effects. Through measurements of real systems combined with new models for estimating P2P traffic, we reveal how much ISPs gain or lose profit due to P2P localization, which indicate that some ISPs will have incentive to subvert P2P localization. (2) **Secure localization services:** how to build the services that provide localization properties while still being able to tolerate malicious participants has not been well understood. My work resulted in an intrusion-tolerant virtual coordinate system that takes advantage of physical laws to withstand advanced attacks. (3) **Secure archi-**

tectures for peer-to-peer streaming: the applications that rely on localization services can also be susceptible to attack. Such is the case for peer-to-peer streaming, a high-bandwidth application, that due to its stringent real-time deadlines is particularly vulnerable. We designed a secure architecture which can continue to provide good performance even while under attack.

We summarize our key contributions:

- We develop models and methodologies by which to understand how P2P localization will effect residential and transit ISPs. We propose a new model to derive P2P inter-AS traffic matrices. Through large-scale measurements of a P2P system and real packet-level traces of P2P traffic from two ISPs we validate the model. We then evaluate how much traffic can an AS actually reduce when localization occurs. Furthermore, we apply BGP inferred AS-paths and the business relations between these ASes to understand how this reduction will affect the profitability of ASes.
- We describe how to protect localization services from insider attacks by identifying invariants in unstructured overlays. We accomplish this by leveraging the physical abstraction that a popular Virtual Coordinate System (VCS) is based on, deriving three invariants from Newton's three laws of motion. We show how to use the three identified invariants to detect and mitigate both well-studied basic attacks, as well as more recent advanced attacks. We conduct extensive simulations and real-world experiments on the PlanetLab testbed [17] to demonstrate that our solution is able to mitigate all known attacks. We also find that, even with no attackers, our solution has better performance than the unprotected VCS, i.e. our solution is 25% more accurate and 68% more stable.
- We present the design of a mesh-based P2P streaming system that can defend against insider attacks. To systematically show how a malicious node could attack the system, we developed a taxonomy of implicit commitments that a node makes with other entities in the system. We then show how, when not explicitly enforced, attackers can break these commitments and attack the

system. Specifically, we consider an important class of attacks where malicious nodes deliberately become neighbors of a large number of nodes and do not upload data to them. We present mechanisms that can enhance the resilience of mesh-based streaming against such attacks. We evaluate our design with real-world experiments on PlanetLab [17] and show that our design is effective. Even when there are 30% attackers, nodes can receive 92% of the data with our schemes compared to 10% of the data without our schemes.

1.3 Thesis Roadmap

The rest of this thesis is organized as follows. A description of our system and attacker model is in Chapter 2. We conduct a detailed simulation study to examine how localizing P2P traffic within network boundaries impacts the profitability of an ISP in Chapter 3. We demonstrate how to secure a localization service, that provides latency estimation, against insider attacks aiming to disrupt them in Chapter 4. We describe a scheme for securing data delivery for high-bandwidth streaming systems that can utilize localization in Chapter 5. We present related work in Chapter 6 and we present our conclusions in Chapter 7.

2 SYSTEM AND ATTACKER MODEL

We now describe our system model for overlay networks, specifically a mesh overlay. We first though describe our model for the underlay that they run on top of, namely the Internet, and also our model for virtual coordinate systems, which provides a localization service for overlays. We also describe our adversarial model by which we study attacks on overlay networks.

2.1 Internet Model

The Internet consists of Autonomous Systems (AS) which participate in the BGP routing protocol, facilitating global connectivity on the Internet. An ISP is made up of at least one, but possibly many, ASes. Each AS has at least one of three business models, which we depict in Figure 2.1. The first model is a content provider, exemplified by Google, who provides services and media to end users. The second model is a residential AS, exemplified by Comcast, who allows end users to access the Internet. The last model is a transit AS, exemplified by Level 3, who interconnects other ASes and carries traffic for them.

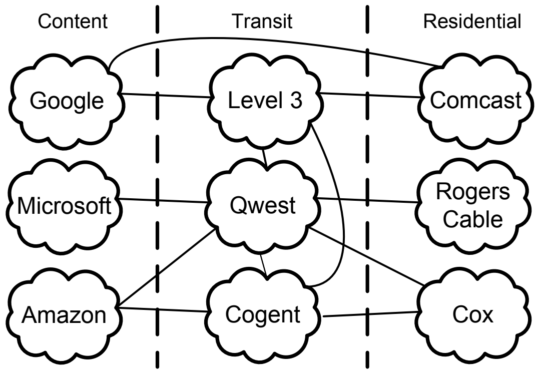


Figure 2.1. ASes and their different business models

As ASes interconnect, they must decide on who must pay who, if at all. Therefore ASes decide if they should form a “peering” relationship, where the ASes carry each others traffic for no cost, or form a “customer-provider” relationship, where the customer pays the provider for traffic. Typically, if a residential and transit AS connect, the former will be a customer of the latter.

2.2 Virtual Coordinate System Model

Virtual Coordinate Systems (VCS) and other localization services similar to them, provide network-awareness to the overlays that run on top of them. VCS accomplish this by embedding the Internet onto a coordinate plane. In a VCS of n nodes, each node will maintain a coordinate value and can estimate its RTT to any other of the n nodes by calculating a distance function on two coordinate values. While the distance function is often Euclidean, the coordinates can also be embedded into a non-Euclidean space such as a hyperbolic space.

Each node maintains information about a small set of m reference nodes. Specifically, it will ping each of these nodes periodically, requesting their coordinate and possibly other auxiliary values. At the same time the node will measure their round-trip time (RTT). These values will allow the node to iteratively update its coordinate through an algorithm specified by the VCS.

2.3 Mesh Overlay Model

An mesh overlay consists of a set of n nodes, where each node maintains a set of neighbor nodes. These neighbors form virtual links, which are unidirectional, and collectively form the topology of the network. An example of a mesh overlay in illustrated in Fig. 2.2. Neighbors are selected based on application-specific criteria, such as the latency or bandwidth that the node can provide. Neighbors that provide data to a node are called in-neighbors , neighbors that a node sends data to are called out-neighbors .

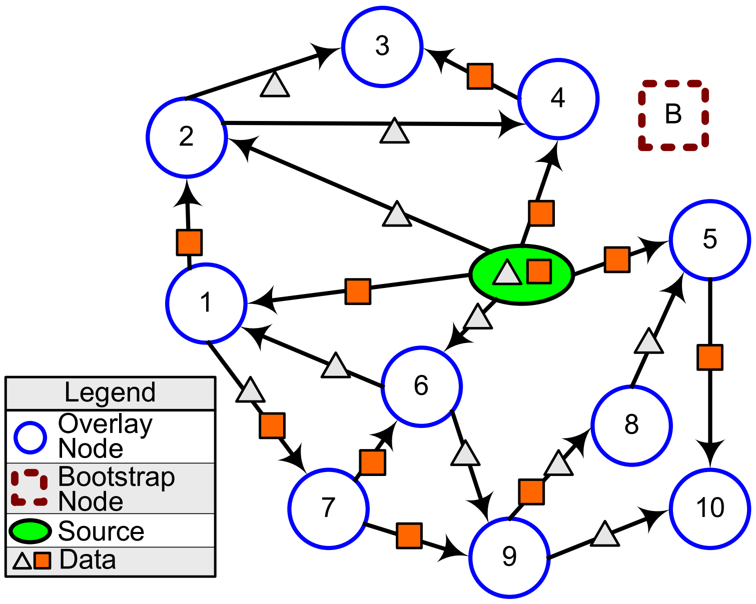


Figure 2.2. Example of a unidirectional mesh-based streaming overlay in which the source sends two different data chunks as denoted by the gray triangle and orange square. Each node has an in-neighbor set and an out-neighbor set. For example, for node 6, the in-neighbor set consists of node 7 and the source, while the out-neighbor set consists of nodes 1 and 9.

To join the overlay network, a node contacts a bootstrap node, which will respond with a subset of nodes already inside the network. The joining node can then establish neighbor links with the reported nodes and participate in the particular protocols of the overlay. We assume that for overlay networks designed for content distribution, there is a trusted source from which data originates. Other than the source, each node has similar functionality and can perform any role that the node needs, such as that of sending or receiving data.

2.4 Attacker Model

We consider insider attackers that have access to all necessary credentials, such as cryptographic keys, to be authenticated into the system and participate in the overlay network. These insiders gained access to keys and possibly stored data through compromising a node or because there is an open access model where anyone can join the system.

While some P2P systems provide incentives so that rational nodes behave well [1] we consider a Byzantine attacker that can behave arbitrarily. This is due to the fact that while faithful participation by rational nodes can be brought about by proper incentives, Byzantine attackers do not necessarily respond to incentives and can often severely affect the performance of the system. We assume that attackers can collude with other attackers. We assume a bounded number of attackers f, where $f < n$.

3 THE INTERNET-WIDE IMPACT OF P2P TRAFFIC LOCALIZATION ON ISP PROFITABILITY

The last decade has seen a rapid growth in popularity of peer-to-peer (P2P) systems, spanning diverse applications such as content distribution (e.g., BitTorrent, eMule, Gnutella), video streaming (e.g., PPLive, Coolstreaming), and audio conferencing (e.g., Skype). However, the success of these applications and the consequent growth in P2P traffic has raised concerns among Internet Service Providers (ISPs), which have to pay a high cost for carrying traffic while receiving little revenue. While there is evidence that P2P traffic is decreasing [18], it still represents today a significant fraction of the Internet traffic (more than 18% according to [18] and more than 50% in some of our datasets), and it is perceived as wasteful of network resources such as expensive peering link bandwidth. In order to reduce these costs, different P2P localization techniques have been proposed [19–25]. The key idea behind these techniques is to limit the amount of traffic entering the ISP by enforcing a preference in exchanging content among peers in the same ISP. In this chapter we examine how localizing P2P traffic within network boundaries impacts the profitability of an ISP.

3.1 Introduction

Several previous works have shown the benefits of localization for both users and providers [19,21], while other works question the possible benefits for users [26]. However, all previous studies consider a partial view of the problem, e.g., by showing the benefits for a single Autonomous System (AS) or running a limited set of experiments involving different ASes. Therefore, it is unclear whether localization is necessarily beneficial to all ASes, how the adoption of localization by one AS impacts other ASes, and how the traffic carried by various ASes is altered as localization techniques are widely adopted.

Evaluating the impact of localization policies when applied on an Internet-wide scale is a challenging task given the complexity of the relationships that exist between different ASes. Specifically, because ASes play various roles from a business point of view, they may experience different effects from the use of localization policies. For example, some ASes (referred to as *residential ASes*), provide Internet service to end-users, and P2P clients are found in these ASes. Other ASes (referred to as *transit ASes*) provide the service of connecting other residential and transit ASes together. However, many transit ASes also provide residential services, and a clean separation between the two types does not exist today. From a business point of view, ASes form "customer-provider" relationships, where a customer AS will pay for the service a provider AS offers, or "peering" relationships, where two ASes will agree to carry each others traffic for free.

Given the current structure of the Internet, localization of traffic is intuitively beneficial for purely residential ASes, and it will have a negative impact on the revenues of purely transit ASes. However, we have found that over 1,200 residential ASes also provide transit service to at least one other AS. Thus, for many ASes it is not obvious how localization may impact them. In addition, as the ultimate goal of ASes is cutting costs and increasing revenue, there are alternative approaches to simply localizing traffic inside an AS, and such approaches have not been explored in previous work. For example, ASes could prefer to exchange traffic with peering ASes. Furthermore, to increase revenue, ASes could prefer to push traffic to customers' ASes and avoid providers' ASes.

Our goal in this chapter is to gain deeper insights into such Internet-wide implications of P2P traffic localization on ISP profits, and develop simulation methodologies to systematically explore the issues. We explicitly focus our work on the benefits and drawbacks for ISPs, though we note that the use of localization can also impact user performance. Our simulations are based on detailed models of (i) inter-AS P2P traffic; (ii) inter-AS routing; (iii) localization policies; and (iv) pricing policies that predict the impact of changes in traffic on ISP profit.

We model inter-AS P2P traffic by leveraging the model proposed in [27], perhaps the only inter-AS traffic model that is available today, in contrast to intra-AS traffic which has been widely studied. We present refinements to the model presented in [27] and show that the refined model has better accuracy. The model requires the knowledge of the P2P population in each AS as input, which we estimate considering BitTorrent, one of the most popular and widely used P2P systems. Our estimation is based on a dataset of over 138 million BitTorrent peers participating in 2.75 million torrents, obtained by crawling a popular tracker. While our evaluations are based on BitTorrent, our methodologies are general, and apply to other P2P systems as well.

Conducting our simulation study requires models that can predict the reduction in P2P traffic entering/exiting an ISP when localization techniques are employed. The possible traffic reduction depends on a wide range of factors including (i) the population of peers inside an AS, (ii) the extent to which peers download similar content, and (iii) the upload capacities of peers inside the ISP relative to those outside [28]. Rather than focusing on a specific localization model, we conduct a sensitivity analysis to a range of models.

As a last step, translating a change in traffic volumes into a change in profits for the ISP is a challenge. While the total profitability of an ISP depends on many factors, such as SLAs, backhaul costs, and private agreements, due to the difficulty of modeling this, we focus only on transit costs. Typical pricing models for transit costs in ISPs are based on the 95th percentile of traffic volumes [29], with the price per Mbps itself showing significant geographical variation. Further, the pricing models depend on total volumes of traffic across all applications rather than P2P traffic volume alone. However, while we are able to estimate P2P traffic volumes, total traffic volumes are unavailable to us. Therefore, we consider multiple pricing models and develop conservative and optimistic predictions of the change in profits for an ISP due to P2P traffic.

Armed with these models, we seek to answer several questions such as: (i) Do ASes necessarily benefit by employing localization? How significant are the benefits? (ii) How is the profitability of various ASes impacted if localization policies are adopted by an increasing fraction of ASes at the same time? What is the impact of global adoption of such policies? (iii) Are there any better policies that can be more profitable to some ASes than a simple localization policy? Given the complexity of the real-world factors that our models seek to capture, there are unavoidable simplifications that must be made. Thus, rather than "absolute" answers to these questions for specific "point-models", our focus is on understanding the sensitivity of our results, and how the trends change with various localization and pricing models.

Our results show that the benefits of localization must not be taken for granted. Some of our key findings include: (i) residential ISPs can actually lose money when localization is employed and some of them will not see increased profitability until other ISPs employ localization; (ii) the reduction in costs due to localization will be limited for small ISPs and tends to grow only logarithmically with client population; and (iii) some ISPs can better increase profitability through alternate strategies to localization by taking advantage of the business relationships they have with other ISPs. Overall, we believe our findings have important implications for ASes, and both our findings, as well as the methodologies and models that we develop in this chapter are important contributions in their own right.

The remainder of the chapter is organized as follows: Section 3.2 introduces our P2P inter-AS traffic model and its validation. Section 3.3 and Section 3.4 discuss different localization policies and the pricing models we use in the chapter. Section 3.5 and Section 3.6 show our findings under different localization scenarios. Finally, main findings of the chapter are summarized in Section 3.7.

3.2 Modeling Inter-AS P2P Traffic

We first describe the model we used to predict an inter-AS P2P traffic matrix. We leverage the gravity model which has been previously used to estimate both intra-AS [30, 31] and inter-AS [27] traffic matrix. As our focus is on inter-AS P2P traffic, we first review the model introduced in [27], which we refer to as the *Gravity* model, then propose a new refinement to improve P2P traffic prediction accuracy, which we refer to as the *Affinity* model.

3.2.1 The Gravity Model

Inter-AS traffic demand has been modeled only once before in the work by Chang et al. [27] which applies the well-established gravity model to an inter-AS setting. The model accounts separately for P2P and web traffic. Below we describe only the P2P component of the model. In the Gravity model, the traffic X_{ij} sent from AS i to AS j is defined as follows:

$$X_{ij} = \frac{f(R_{RA}(i))f(R_{RA}(j))}{R_{BA}(i,j)^{\beta}}, \tag{3.1}$$

where f is the monotonically decreasing function $f(x) = 1/x$, $R_{RA}(i)$ is the rank of AS i in the list of ASes sorted by decreasing peer population, and $R_{BA}(i,j)$ is the rank of the bottleneck AS between i and j in the sorted list of ASes by capacity (the bottleneck AS is the smallest transit AS that is on the AS path between i and j). This model stems from the intuition that the higher the population of peers in an AS (i.e., the higher is its rank), the larger the aggregate of traffic the AS exchanges. In addition, if the path between two ASes has little capacity, then the amount of traffic will be consequently reduced. β is a parameter that is used to better weight the effect of bottlenecks along the path. In [27] $\beta = 0.1$ is suggested, which makes the bottleneck bias almost negligible. This implicitly suggests that the volume of P2P traffic exchanged between ASes is mainly driven by the peer population of each AS.

3.2.2 The Affinity Model

Given the world-wide nature of the Internet and its diversity of users and available content, intuition suggests that P2P traffic will be driven not only by the population size of ASes, but also by the cultural and linguistic makeup of the users inside the ASes, or the "*affinity*" between ASes. Thus, if peers of two ASes are not interested in the same content, the traffic exchanged among them will be marginal, even if the number of peers in each AS is large. For example, if AS-1 and AS-2 are located in Italy, and AS-3 is located in China, it is expected that large traffic will be exchanged between AS-1 and AS-2, while little traffic will flow between AS-3 and AS-1, AS-2.

We augment the Gravity model to also account for the affinity between ASes. We estimate the affinity between ASes using the cosine similarity distance [32]. The cosine similarity results in a value between 0 (no similarity) and 1 (perfect similarity) that is the cosine of the angle between two vectors $\bar{V_i}$ and $\bar{V_j}$, i.e.,

$$Cos(i,j) = \frac{\bar{V_i} \cdot \bar{V_j}}{\|\bar{V_i}\|\|\bar{V_j}\|}. \tag{3.2}$$

In our case, each vector $\bar{V_i}$ represents the "content distribution" in AS i, whose components report the number of peers interested in a given content that are present in AS i. Thus, if two ASes have many peers interested in the same content, then they will have high affinity. Completing the previous example, consider as content an Italian movie, a Chinese song, and an English book. Assuming $\bar{V_1} = (10, 1, 3)$, $\bar{V_2} = (100, 2, 30)$ and $\bar{V_3} = (0, 10, 3)$, we have $Cos(1, 2) = 0.997$ while $Cos(1, 3) = 0.173$, which reflects the intuition that Italian ASes prefer to exchange traffic among themselves rather than with the Chinese AS.

Once we have calculated the affinity between two ASes we can combine it with the peer population in each AS to form a gravity model. Thus, we define our Affinity model as follows:

$$X_{ij} = P(i)P(j)Cos(i,j), \tag{3.3}$$

where $P(i)$ and $P(j)$ are the population of AS i and AS j.

While the Affinity model could include a preference related to the upload capacity of peers, we chose to include only the affinity among ASes due to client's interest in the same content. We superpose a bias in peer selection due to performance as part of the locality models in Section 3.3.2.

3.2.3 Model Validation

We first describe the datasets we use as input to the Affinity model and also use throughout the chapter. We then present results that compare our model with the Gravity model.

BitTorrent crawl snapshots: The Affinity model requires as input the peer population $P(i)$ and the content distribution vectors $\bar{V}_i$. To estimate them, we rely on active measurements obtained by crawling a very popular BitTorrent tracker named "*OpenBitTorrent*" [33]. As the tracker is not associated with a particular torrent publishing web site and it provides an easy way for users to publish content, it attracts users from all over the world.

We took snapshots of BitTorrent activity every hour for a period of 8 days during May 2010. A total of 192 different snapshots have then been collected, which will be used throughout this chapter. In each snapshot, we crawled all torrents that had at least one active downloader and for every torrent we requested peers from the tracker until we received at least 95% of all participating peers. Since many users are behind NATs, we consider a peer to consist of a unique (IP, port) combination. This allows us to obtain information about which peers are actually participating in which torrent, i.e., the peer population by content. To obtain the peer population per AS, we map IP addresses to the corresponding AS by using the service provided by Team Cymru [34]. At the end, for every snapshot we obtain for each AS i the population $P(i)$ and content distribution $\bar{V}_i$, which allow us to compute 192 global AS level traffic matrices.

While a detailed characterization of these BitTorrent datasets is out of the scope of this chapter, we briefly summarize their size which reflects their generality. A normal snapshot consists of over 5 million peers, 154 countries, 12,000 ASes, and 1 million torrents. Over the 8 days, we saw more than 138 million distinct peers in over 2.75 million torrents. One interesting finding about the dataset which will be instrumental later is the fact that each torrent population size follows a heavy tailed distribution with a small portion of very large torrents, but also a large number of torrents with less than 100 peers. Peer distribution over ASes is instead more biased toward larger ASes which host most of the peers, e.g., the largest 1,600 ASes account for 97% of peers.

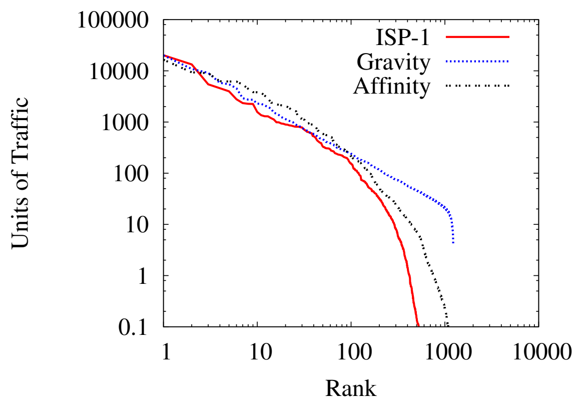
Inter-AS topology and routing: The knowledge of the AS paths is instrumental to predict the volume of traffic on individual inter-AS links. Besides, they are also necessary to compute $R_{BA}(i)$ for the Gravity model. First, we need a map of the AS topology which includes the business relationships between ASes. We use CAIDA's AS map [35] augmented with peering edges from recent research on mapping Internet Exchange Points (IXPs) [36]. Second, we need to know the AS-level routing. To this end, we use the algorithm proposed by Qiu et al. [37] to determine valley-free paths between residential ASes. Qiu et al.'s algorithm uses Routing Information Bases (RIBs) alongside the AS topology to determine the most likely route between ASes. We use RIBs provided by the Oregon's RouteViews Project [38] that are from the LINX, KIXP, PAIX, and Equinix Ashburn IXPs. This set of routing table dumps represents over 329,000 prefixes from 33,910 ASes.

Leveraging on the fact that the largest 1,600 ASes according to peer population alone account for 97% of all the P2P traffic that is generated by the Affinity model, we limit our evaluation to only this subset. We also include all the transit ASes that belong on any AS path between these residential ASes, for a total of 2,067 ASes. In this chapter, we define a *residential* AS as having at least one peer in the BitTorrent crawl and a *transit* AS as having at least one customer AS in the CAIDA map. Thus, an AS could be both a residential and a transit AS. More details about the ASes are deferred to Sec.3.5.

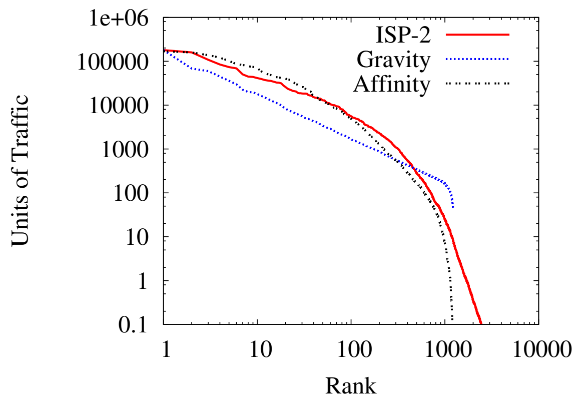
Packet traces from large ISP datasets: To verify the accuracy of the Affinity model traffic prediction, we compare its output against packet-level traces from six vantage points scattered in the US and across three different European countries. Each vantage point monitors several thousands of users. For convenience, we name the vantage points ISP-1 to ISP-6. For each ISP, all packets going to and coming from all the hosts in the Points of Presence (PoP) were passively monitored for several months. An advanced traffic classification tool based on deep packet inspection and advanced statistical classifiers [39] was used to produce the per-application volume of traffic sent by hosts in the PoP to each different AS, i.e., an actual row of the traffic matrix for each given application.

We show results from comparing models in Fig. 3.1, where we focus on a one-day long trace from ISP-1 and a one-hour long trace from ISP-2. We use the BitTorrent snapshots that refer to the same time of day that the ISP traces are from. Similar results were obtained for the rest of the traces.For each graph, we report the volume of traffic sorted in decreasing order, considering actual measurements (solid line), the Gravity model prediction (small dot line), and the Affinity model prediction (large dot line). As both the Gravity and Affinity models produce unit-less output, we scale them and the ISPs' measured traffic volumes so they are comparable to a standard unit-less metric by minimizing the mean square error. We also show the corresponding relative error values of both models in Fig. 3.2.

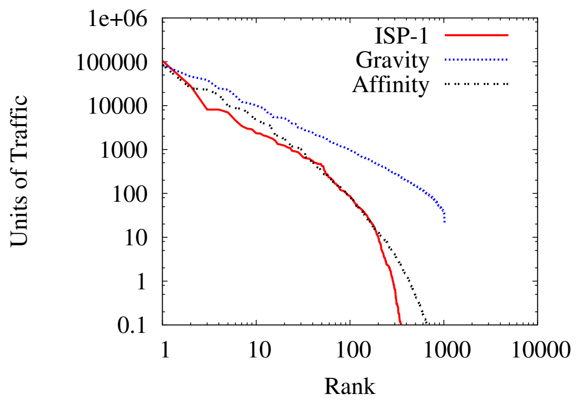
Fig. 3.1(a) refers to BitTorrent traffic as seen from ISP-1. The Gravity and Affinity models are very similar until rank 300, at which point the Gravity model severely overestimates the traffic demand, while the Affinity model better captures the sudden decrease of traffic sent by ISP-1 clients to the smaller ASes. Similarly, we compare BitTorrent traffic seen from ISP-2 in Fig. 3.1(b). Again, the Affinity model is able to better match the traffic demand trend for most ASes, while the Gravity model shows a much more regular slope, clearly missing the content bias induced on exchanged



(a) BitTorrent dataset from ISP-1.

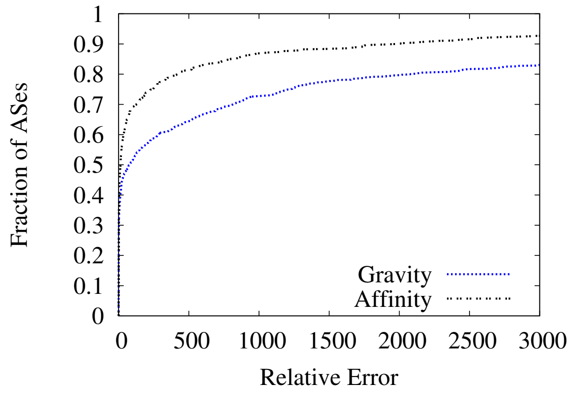


(b) BitTorrent dataset from ISP-2.

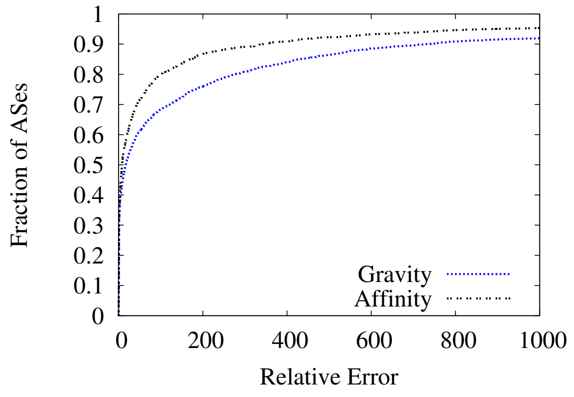


(c) eMule dataset from ISP-1.

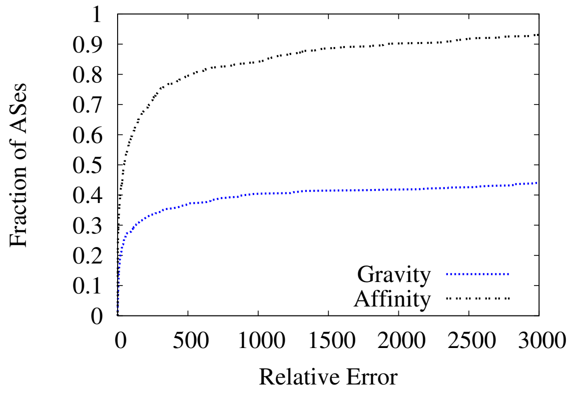
Figure 3.1. Affinity and Gravity models compared against real measurements.



(a) BitTorrent dataset from ISP-1.



(b) BitTorrent dataset from ISP-2.



(c) eMule dataset from ISP-1.

Figure 3.2. Relative error for Affinity and Gravity models

traffic. For the relative error values in Fig. 3.2(a) and 3.2(b), the Gravity model is only very accurate for 50% of ASes, while the Affinity model is accurate for 70% and 60% of ASes, respectively.

Finally, to show that the Affinity model is not specific to BitTorrent but can be generally applied to other P2P protocols, Fig. 3.1(c) shows results considering traffic volumes sent by ISP-1 clients, but using eMule as the P2P application. The same cosine similarity values as obtained from the BitTorrent snapshots are used, since the cosine similarity values catch the cultural and linguistic interests of peers, and are not expected to change across different P2P systems. The per AS eMule population has been estimated from the eMule traffic in ISP-1 instead. Also in this case, results show that the Gravity model overestimates the actual traffic sent to each AS, while the Affinity model closely matches the traffic demand even up to high ranking values. This difference is seen in Fig. 3.2(c), where the Gravity model is only very accurate for 30% of ASes, but the Affinity model is accurate for 60% of ASes.

We have conducted other experiments to verify the goodness of the Affinity model, considering different times of the day, different days, transmitted and received traffic, different P2P systems and different crawls from different trackers. In all cases, the Affinity model provided more accurate estimates than the Gravity model. Moreover, the cosine similarity proved to be very robust, so that it can be used to model several P2P applications like BitTorrent or eMule.

3.3 Modeling P2P Localization

In this section we present models to predict the reduction in P2P traffic exchanged by an ISP if localization techniques are employed. We are not attempting to model particular P2P systems in this section, but simply what could happen if localization occurs.

A single model may not be sufficient because P2P traffic reduction depends on a variety of factors such as (i) the population of peers inside an AS, (ii) the extent to which peers download similar content, and (iii) the upload capacities of peers inside the ISP relative to those outside [28]. Hence, we consider a set of locality models and show sensitivity to them. Validation of these models is a difficult task, since this requires measuring P2P traffic aggregates from a large number of ISPs around the world, from different ISP categories (e.g., residential and transit), and with different upload capacities of clients. Instead, in later sections, we show trends of the impact that P2P localization may have on ISPs and perform extensive sensitivity analysis to the various localization policies.

For each model we determine the ratio of traffic received by an AS j after localization versus before localization, which we call α_j. In other words, α_j is the *fraction of leftover traffic* after localization that still will be received by peers in AS j. Intuitively, a good localization policy will result in a small α_j value. The traffic L_{ij} sent by peers in AS i to peers in AS j after localization is then simply:

$$L_{ij} = \alpha_j X_{ij}, \tag{3.4}$$

where X_{ij} is the traffic demand generated by the Affinity model in Equation 3.3. As we have multiple snapshots from which we generate traffic matrices, we also calculate α_j for each snapshot. For simplicity though, we drop the explicit notation on time in the following.

3.3.1 System Architecture Assumptions

We assume that there is a localization technique in place that allows peers to find other peers that are in the same AS. For example, peers contact an “oracle” which allows them to obtain an ordered and possibly filtered list of peers interested in the same content. Peers then start exchanging data with the suggested peers according to the P2P trading algorithm. Individual ASes can impose localization of traffic independently of what other ASes do, e.g., some may deploy an oracle, others may not. This scenario is compatible with both the P4P iTracker [19] and the IETF ALTO [40] proposals.

We further assume that an AS cannot influence peers outside of its own AS, so that external peers can still connect to and download from internal peers, i.e., an AS cannot stop external peers from downloading content from peers within the AS. This implies that transit ASes do not deploy traffic shaping on traffic that does not originate from their own AS, but only rely on the oracle to enforce localization policies. Furthermore, this implies there is some altruism in the system, so that clients in an AS that do not localize traffic can still receive the content, even if every other AS does localize traffic. Therefore, for an AS that does localize, its outgoing P2P traffic can be greater than its incoming traffic.

3.3.2 Locality Models

• **Single(no history):** This model captures a pessimistic scenario where for every crawl, the file must be downloaded again by every peer from outside the AS. That is, there are no internal seeds available.

The model computes the leftover traffic assuming only *one single copy* of the content will need to be downloaded from outside the AS. Once the initial copy has entered the AS, content will be exchanged only among local peers. For example, assume there are $P_j(k) = 10$ peers from AS j downloading content k; when localization is used, only one copy would need to be downloaded, resulting in $1/P_j(k) = 0.1$ leftover traffic. Thus, the more popular a piece of content is, the less leftover traffic there will be. Given a snapshot, for every AS j that has clients in N_j distinct torrents, we estimate α_j as follows:

$$\alpha_j = N_j \frac{1}{\sum_{k=1}^{N_j} P_j(k)}. \tag{3.5}$$

• **Single(history):** This model captures a more realistic model where we consider that the first time a peer appears in a torrent in our crawls, it is considered a leecher, and if it appears again in that torrent in later crawls we consider it to be a seeder. To find out how sensitive α_j actually is to content availability, we simply keep track of what peers have been in which torrents over time. For a given torrent, consider a peer that has been seen at time slot t for the first time. When it reappears in time slot $t' > t$, it is considered a seed. That is, if the peer has been in a torrent in the past, it is marked as a seed in future time slots. Formally, for a snapshot t and AS j, let $S_j(k)$ be the number of seeds in torrent k and let T_j be the number of torrents that have some seed in them. We can then calculate α_j with the following equation:

$$\alpha_j = \frac{N_j - T_j}{\sum_{k=1}^{N_j}(P_j(k) - S_j(k))}. \tag{3.6}$$

• **Single(persistent):** This model represents an optimistic scenario, where once a single peer inside an AS downloads a file, then no other peer inside the AS will need to download from outside the AS again, since the initial peer remains as a seeder for everyone else. Thus, content k is made available to local peers forever after it has been downloaded once from the outside at time slot t. We use Equation 3.6 to calculate α_j for this model, but assume at least one seed is always present for each time slot $t' > t$.

• **Perf(no history):** We also consider policies that *include a performance bias* since peers might prefer to download from nodes with a higher upload capacity than those inside its AS. The first performance model captures the scenario when a peer prefers to download content from peers in its own AS, unless there exists external peers with much higher upload capacity. A similar policy has been examined in [28]. We compute $E(j,k)$, the expected number of copies of content k downloaded from outside AS j.

$$E(j,k) = P_j(k)\frac{U(j,k)}{P(k)}, \tag{3.7}$$

where $P(k) = \sum_j P_j(k)$ is the total number of peers interested in content k, and $U(j,k)$ is the number of external peers interested in content k that have an average upload capacity higher by a factor of γ than peers in AS j. By averaging over all content in which AS j participates we have:

$$\alpha_j = \frac{\sum_{k=1}^{N_j} \max(E(j,k),1)}{\sum_{k=1}^{N_j} P_j(k)}, \tag{3.8}$$

where $\max(E(j,k),1)$ states that at least one copy must be downloaded.

For the evaluation of this scheme, we use the iPlane [41] dataset, which provides an estimate of the access bandwidth of several tens of thousands of /24 networks in the Internet. To account for factors that could make the real and the estimated capacities differ, such as congestion of intermediate links, we select a remote peer over a local peer only if the access bandwidth of the remote peer is at least 10 times higher than the bandwidth of the local peer. Furthermore, any remote peer for which we do not have access bandwidth information will not be preferred over a local peer.

• **Perf(history):** Similar to the previous model, if an internal seed exists at time t, then peers do not need to download anything from outside the AS. The following equations are used to calculate α_j:

$$E(j,k) = (P_j(k) - S_j(k))\frac{U(j,k)}{P(k)} \tag{3.9}$$

$$\alpha_j = \frac{\sum_{k=1}^{T_j} E(j,k) + \sum_{k=T_j+1}^{N_j} \max(E(j,k),1)}{\sum_{k=1}^{N_j}(P_j(k) - S_j(k))}, \tag{3.10}$$

• **Perf(persistent):** We again assume that content persists forever after being downloaded once from outside the AS. We use Equation 3.10 to calculate α_j for this scenario, but assume a seed always persists after the first download.

3.4 Measuring ISP Profitability

The total profitability of an ISP depends on many factors. Due to the difficulty of accurately modeling all the costs associated with carrying traffic, such as backhaul costs, we do not attempt to do so. In this chapter we focus on the portion of the profits/expenses that are related to money gained/paid due to the transit costs of carrying P2P traffic only. We define our ideal metric for achieving this goal and describe how we evaluate it using our pricing models.

3.4.1 An Ideal Metric for ISP Profitability

A customer ISP i typically gets charged by a provider ISP j based on the 95th percentile (P95) volume of traffic exchanged on an individual link [29]. This is done by sampling the inbound and outbound volume of traffic every 5 minutes for the duration of a billing period, which is usually 30 days. Let $V_{ij}(t)$ and $V_{ji}(t)$ respectively denote the outbound and inbound volumes for ISP i at time instant t. After sorting these values, P95 is chosen from both the outbound and inbound traffic; let these terms be denoted as $P95(V_{ij})$ and $P95(V_{ji})$. Let CV_{ij} be the charging volume, which is the actual volume that charges are computed on. Typically, CV_{ij} is determined by taking the maximum of the inbound and outbound P95s:

$$CV_{ij} = \max(P95(V_{ij}), P95(V_{ji})). \tag{3.11}$$

Alternatively, while not widely used, in some cases it is determined by taking the average:

$$CV_{ij} = (P95(V_{ij}) + P95(V_{ji}))/2. \tag{3.12}$$

CV_{ij} is then used as input to a pricing function, which is typically non-decreasing, the output of which is a dollar amount that the customer owes the provider. Assuming a linear pricing function (see Sec 3.4.3 for more details), let p_{ij} be the price per Mbps for the link between i and j, then the amount that ISP i owes ISP j is $p_{ij}CV_{ij}$.

Let $\mathcal{P}_i$ and $\mathcal{C}_i$ denote the set of providers and customers of ISP i, respectively. Then, the profit of the ISP i prior to localization is $\sum_{k\in\mathcal{C}_i} p_{ik}CV_{ik} - \sum_{k\in\mathcal{P}_i} p_{ik}CV_{ik}$.

Thus far we have considered the profit with respect to a certain set of traffic volumes. However, if these traffic volumes change due to P2P localization policies, we can also calculate the increase in profits after this occurs. Formally, let $\delta(x)$ denote the change in a variable x when localization is employed. Then,

$$\delta(\text{profit}) = \sum_{k\in\mathcal{C}_i} p_{ik}\delta(CV_{ik}) - \sum_{k\in\mathcal{P}_i} p_{ik}\delta(CV_{ik}). \tag{3.13}$$

To study how localization affects profit due to P2P traffic, we normalize $\delta(profit)$ to the profit before localization that is attributed to P2P traffic ($profit_{p2p,before}$), i.e., profit is computed exactly as before, except only the portion of traffic that is P2P is considered. Thus, we have:

$$\text{profit increase} = \frac{\delta(profit)}{profit_{p2p,before}}. \tag{3.14}$$

Finally, if $profit_{p2p,before}$ is negative (i.e., the ISP is originally losing due to P2P traffic), we simply normalize by the loss instead of the profit. Thus, if $loss_{p2p,before} = -profit_{p2p,before}$, we have:

$$\text{loss reduction} = \frac{\delta(profit)}{loss_{p2p,before}}. \tag{3.15}$$

3.4.2 Approximating ISP Profitability

Ideally, Equation 3.13 should be evaluated considering the total amount of traffic flowing across links. Unfortunately, modeling total inter-AS traffic is a hard problem. To the best of our knowledge, only [27] addressed this problem. However, that model is not easily applicable to our context as it assumes the ratio of P2P to other traffic is known for all ASes which varies widely and is difficult to ascertain.

To handle this, we approximate the ideal metric by assuming that the change in P95 of total traffic volume on localization is the same as the change in P95 of P2P traffic volumes on localization, in each of the inbound and outbound directions. Formally, let $V_{p2p,ij}(t)$ and $V_{p2p,ji}(t)$ respectively denote the inbound and outbound volumes of P2P traffic that ISP i sends to or receives from ISP j at time instant t. Then, we assume that $\delta(P95(V_{ij})) = \delta(P95(V_{p2p,ij}))$, and $\delta(P95(V_{ji})) = \delta(P95(V_{p2p,ji}))$. With this assumption, the approximate change in charging volume on localization is simply computed as follows: (i) if charging volumes are computed based on the maximum, as in Equation 3.11, then $\delta(CV_{ij}) = \delta(P95(V_{p2p,ij}))$ or $\delta(P95(V_{p2p,ji}))$, depending on whether the AS is inbound dominated or outbound dominated; and (ii)

Table 3.1
Approximating the 95th percentile for incoming traffic

Trace	Real [Mbps]	Approximation [Mbps]	Relative Error [%]	P2P Traffic [%]
ISP-1	1221.8	1247.5	2.10	48.6
ISP-2	1782.7	1660.9	6.83	45.1
ISP-3	1053.1	1029.6	2.23	53.02
ISP-4	1845.6	1765.3	4.35	42.5
ISP-5	1385.7	1347.1	2.79	50.4
ISP-6	1350.6	1173.6	13.1	6.5

if charging volumes are computed based on the average, as in Equation 3.12, then $\delta(CV_{ij}) = (\delta(P95(V_{p2p,ij})) + \delta(P95(V_{p2p,ji})))/2$.

Intuition suggests that the daily traffic periodicity is due to human habits. During the day, more users are connected to the Internet and traffic grows. There is thus a correlation between the time at which the P95 happens and the time at which most users are online. For P2P traffic, users run P2P applications when they are online. It is thus likely that the P95 of total traffic happens closely to when the P95 of P2P traffic is reached [42]. Table 3.1 compares the P95 on the inbound traffic observed on the different ISP traces described in Sec. 3.2.3 over a one-week long period of time. The second column shows the actual P95 of total traffic while the third column shows the total traffic observed at the time when the P95 of P2P traffic occurs. The fourth column reports the relative error and the fifth column reports the percentage of P2P traffic from the total traffic. As can be seen, the relative error ranges between 2% to 13% depending on the ISP link. Furthermore, the larger the fraction of P2P traffic in the monitored ISP, the smaller the relative error. We repeated the analysis for outbound traffic. The relative error was even smaller since the fraction of outbound P2P traffic was higher than 80% for all ISPs and thus P2P traffic dominates the P95.

Second, as we have seen that P2P traffic volumes prior to localization tend to be correlated to total traffic volumes, we now argue that the trend will continue after localization. We have found in our datasets that the ratio of P2P traffic after localization to P2P traffic before localization does not vary much over time for all links, and locality models. Thus, it is reasonable to assume that the time the P95 occurs after localization does not shift. For instance, when the Single(history) locality model is used, for all links, the standard deviation of the ratio across various time snapshots (σ) is very small. In particular, 58% of links have $\sigma < .05$ and 90% have $\sigma < 0.1$.

Overall, the discussion above suggests that the errors introduced due to our approximations will be limited in practice, and our prediction of the impact of localization on ISP profitablity will be reasonable.

3.4.3 Pricing Models

We now discuss the models we use to compute the pricing function, and charging volumes. While pricing functions are often non-decreasing piece-wise linear [29] they are specific to each provider and require the total volume of traffic to be known. In order to facilitate our evaluation we assume ASes use linear pricing functions where the charging volume is multiplied by the unit traffic volume price. Linear pricing functions are a good first step towards finding the actual costs and have also been used in determining transit costs for content providers [43]. Linear pricing is a valid approximation in our case because the reduction/increase of traffic that is experienced due to localization policies is not so large to trigger an economy of scale range change in the pricing. Moreover, assuming linear pricing corresponds to evaluating an upper bound on the possible savings an AS can achieve given the sublinear effect induced by economy of scale.

Table 3.2
Pricing functions

Geographic Location	$ per Mbps
North America	10
Europe	14
Australia	34
Asia	38
South America	76

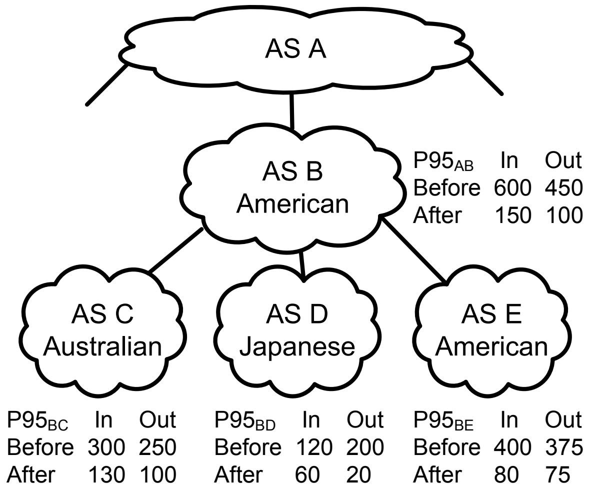


Figure 3.3. Example topology illustrating our pricing model. P95 refer to P2P traffic.

As the price per Mbps is known to vary widely due to geographic location [44] we gather data from Telegeography Research [45] (summarized in Table 3.2) to determine how customer ASes are charged.

We next discuss our models for charging volume, using Fig. 3.3 to aid our discussion. Assume that ASes B, C, D, and E are all residential ASes. The P95s of P2P traffic for all links before and after localization are reported in the figure. For instance, on the link between A and B, the P95 of the P2P traffic inbound to B

is 600 Mbps and 150 Mbps, before and after localization respectively. Likewise, the P95 of the P2P traffic outbound from B is 450 Mbps and 100 Mbps, before and after localization respectively.

We now summarize the various pricing models we use:

• **Average:** For the charging volume we calculate the average of the inbound P95 (P95$_{IN}$) and outbound P95 (P95$_{OUT}$) for each link as in Equation 3.12. The change in charging volume on localization may be approximated as in Sec. 3.4.2. For instance, in Fig. 3.3, the charging volume on the link between B and A would decrease from $(600 + 450)/2$ Mbps to $(150 + 100)/2$ Mbps, a reduction of 400 Mbps. Considering traffic prices from Table 3.2, AS B is charged \$10 per Mbps by A. Thus, localization will reduce B's costs by $\$4,000$. However, the charging volume will also reduce on links from B to each of its customers, resulting in revenue reductions. The revenue reduction is $34*(300+250)/2-34*(130+100)/2 = \$5,440$ from customer C, $38*(120+200)/2-38*(60+20)/2 = \$4,560$ from customer D, and $10*(400+375)/2-10*(80+75)/2 = \$3,100$ for customer E. The $\delta(profit)$ for B is then $-\$9,100$ and the profit increase is $\delta(profit)/profit_{p2p,before} = -\$9,100/\$14,055 = -0.65$, indicating that 65% of profits on P2P traffic were lost.

• **Upper and Lower Bounds:** In contrast to the average case, computing changes in charging volume is more complicated if the pricing scheme is based on the maximum of P95$_{IN}$ and P95$_{OUT}$, as in Equation 3.11. Using such pricing schemes requires us to know whether the total traffic volume is higher in the inbound or outbound direction. However, we only have information regarding P2P traffic volumes. It is possible that P2P traffic volumes are higher in the inbound (outbound) direction, while total traffic volumes are higher in the outbound (inbound) direction. We address these challenges by computing instead an upper and lower bound of the benefits that localization could have on each ISP.

Consider again the link between A and B in Fig. 3.3. Depending on whether B is charged based on inbound or outbound traffic prior to localization, and allowing for a change in the direction of charging volume after localization, the reduction in charging volume may range between $450-150=300$ Mbps, and $600-100=500$ Mbps. While precise determination of the change in traffic volume is difficult, the best possible scenario for B is a reduction of 500 Mbps, while the worst scenario is a reduction of 300 Mbps. More generally, for a customer, the best possible case is obtained assuming $\max(P95_{IN}, P95_{OUT})$ before localization, and $\min(P95_{IN}, P95_{OUT})$ after localization. For a provider the opposite set of choices provides the best scenario. We also observe that on any link, the best scenario for the provider is the worst scenario for the customer, and vice versa. To compute the upper (lower) bound in terms of benefits for an AS when localization policies are applied, we assume the best (worst) case for each of its links.

We now illustrate the lower and upper bound computation for B. In the worst case scenario, the decrease in costs on provider links on localization is $10*(450-150)=\$3,000$, while the decrease in revenue from customers is $34*(300-100)+38*(200-20)+10*(400-75)=\$16,890$. Thus, the lower bound on $\delta(profit)$ is $-\$13,890$ and profit decrease is 80%. However, in the best case scenario for B, the decrease in costs on provider links on localization is $10*(600-100)=\$5,000$, while the decrease in revenue from customers is $34*(250-130)+38*(120-60)+10*(375-80)=\$9,310$. Thus, the upper bound on $\delta(profit)$ is $-\$4,310$ and the profit decrease is 37%.

• **Class:** Since knowing if an AS link is inbound or outbound dominated for all links is practically impossible, we consider a scenario that we build to be as realistic as possible. We use PeeringDB [46], which is a database where network operators document information in hope of attracting other ASes to peer with. The database contains over 1,900 ASes that provide the ground truth by labeling themselves as having traffic ratios that are dominated by inbound, outbound, or are balanced. About 500 ASes are in our dataset and for them we explicitly consider this information.

For the remaining ASes, we hypothesize that the ratio of P2P client to web server populations has a large impact on the amount of traffic entering and leaving an AS. This is because we would expect a residential AS with many P2P clients to have a large number of users consuming content; hence a large amount of inbound traffic. On the other hand, we would expect an AS hosting many web servers to have large outbound traffic. To discover the server population per AS we use a methodology similar to that used in [27] and find 1 million servers in 19,000 ASes. We use then PeeringDB as ground truth to calibrate the threshold ratio to classify ASes. Indeed, we do find a strong correlation between dominating traffic direction and the ratio of population sizes. Considering the unclassified ASes, we find 95% have population ratios clearly indicating they are inbound dominated (and we label as such in this scenario); this is unsurprising as we would expect most ASes in our dataset to be residential ASes and not content providers. Given each AS classification, the corresponding P95 of incoming or outgoing traffic will be used as the charging volume for every provider link the AS has.

To complete the example, let ASes B, C, D and E be classified inbound dominated. Then, the decrease in costs for AS B is $10 * (600 - 150) = \$4,500$, while the decrease in revenue is $34 * (300 - 130) + 38 * (120 - 60) + 10 * (400 - 80) = \$11,260$. This translates to a $\delta(profit)$ of $-\$6,760$ and a profit decrease of 68%.

3.5 Impact of Localization Policies

In this section, we evaluate the profitability of ISPs according to the P2P traffic that they carry today and how localization will affect it. We consider scenarios where a different fraction of ASes localize traffic as ISPs may implement locality policies independent of one another. We also perform sensitivity to locality models and pricing models.

Using the Affinity model and the BitTorrent crawl, we consider, for each locality model, a set of 168 traffic matrices derived from the last 7 days of the 8 day long crawl. The first day is not used in order to discard initial transient conditions for the history and persistent locality models. For each matrix, traffic is then routed on the AS level topology using the AS paths inferred as described in Section 3.2.3. Finally,

Table 3.3
ASes profiting or losing by category (no localization)

AS Type	# Profiting (%)	# Losing (%)
All ASes	322 (16%)	1745 (84%)
Stub	60 (5%)	1140 (95%)
Small ISP	115 (20%)	458 (80%)
Large ISP	139 (49%)	147 (51%)
Tier-1	8 (100%)	0 (0%)

for each customer-provider link, the P95 of P2P traffic is computed considering the 168 samples.

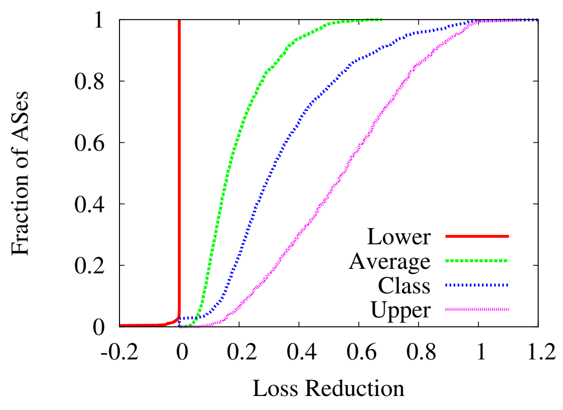
So far we have classified ISPs based on their customer-provider relationship as transit or residential to allow us to clarify the implication of the pricing model. However, this classification does not capture the implication of the AS size on the ISP's profitability, therefore we also categorize each AS according to how many downstream customers it has as proposed by the Internet Topology Collection [47]. There are four categories: Stub, Small ISP, Large ISP and Tier-1, which intuitively state how big a transit AS is. Stubs have less than 5 downstream customers, Small ISPs have 5 or greater, but less than 50, and Large ISPs have 50 or greater. Tier-1 ISPs are those who have no or very few providers and are the same as those identified by [47]. In our dataset there are 1200 Stubs, 573 Small ISPs, 286 Large ISPs, and 8 Tier-1 ISPs.

3.5.1 Profitability before Localization

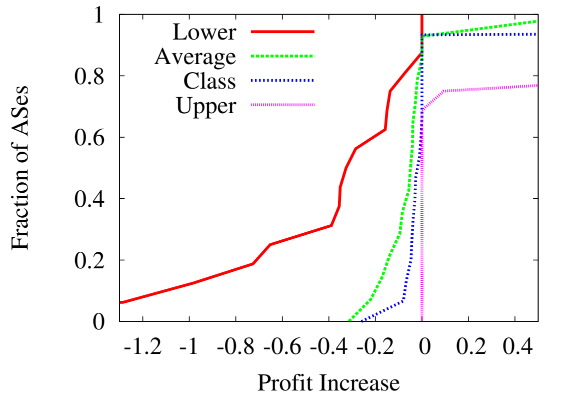
We first consider the scenario where there is no localization used on the Internet. We determine for each category of AS, the number of ASes that are profiting or losing from carrying P2P traffic. We present results only for the Class pricing model since results for Average are similar and Upper and Lower can be calculated only when

Table 3.4
Interpreting metric results

Value	**Loss Reduction**	**Profit Increase**
less than -1	more loss	turned to loss
between -1 and 0	more loss	less profit
between 0 and 1	less loss	more profit
greater than 1	turned profitable	more profit



(a)



(b)

Figure 3.4. Individual AS deploys localization with Single(history) locality model. Sensitivity to pricing models.

localization occurs. Table 3.3 reports the results. As expected, the vast majority of ASes lose money because of P2P traffic (see the first line summary). However, as the number of downstream AS customers increases, there are ASes that profit due to P2P traffic. Indeed, 322 ASes (16%) today are profitable overall, of which 266 ASes are residential. This indicates that not all ASes may want to limit P2P traffic, and only ASes that have few customers have the most incentive to limit external P2P traffic.

Considering ASes that have losses due to P2P traffic, over 51% of them are purely residential, serving end users but not carrying traffic for other ASes. Surprisingly, several ASes that have more than 500 downstream customers still suffer losses. Investigating further, we found that their relationship to Tier-1 ASes largely determines whether they profit or lose. Being a customer of a Tier-1 AS makes the AS lose money, while those that have peering agreements made a profit.

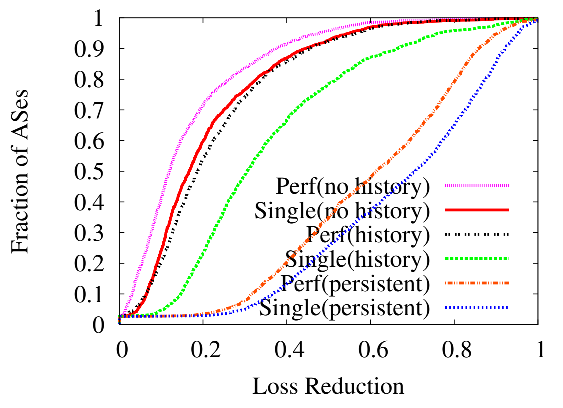
A closer look reveals that some ASes are still profitable, in spite of having few provider agreements with Tier-1. For example, the 13th largest profitable AS (AS-12956, Telefonica) has few agreements with Tier-1 ASes but more than 500 downstream customers, most of which are in Spanish speaking regions. By carrying mostly traffic that is exchanged among South American and other Spanish ASes, it takes advantage of the cultural and linguistic characteristics of P2P inter-AS traffic to send high volumes of profitable traffic between customer ASes and very little costly traffic to Tier-1 providers.

Insight #1: *Transit ASes that have customer ASes with similar cultural and linguistic makeups benefit more from carrying P2P traffic than those whose customer ASes are dissimilar. A transit AS with such customer ASes sends more traffic to customers than to providers, increasing its revenue.*

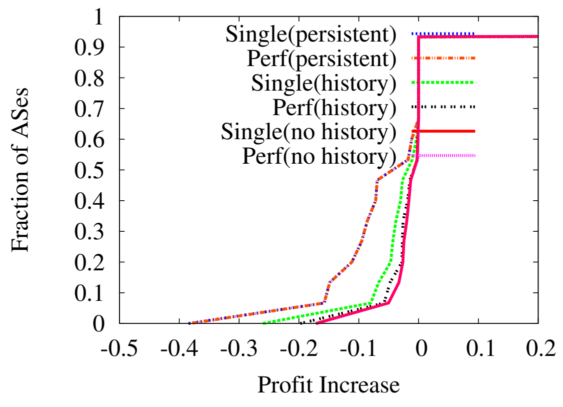
3.5.2 Localization Deployed by Individual ASes

We seek to understand if localization is beneficial for an individual AS, independent of what other ASes do. Specifically, we investigate what is the expected benefit for an AS that deploys a localization policy alone. We consider only residential ASes since pure transit ASes have no benefits in localizing traffic (having no clients).

Sensitivity to pricing model: We fix the locality policy to Single(history) and calculate the charging volumes as described in Section 3.4.3. We use the metrics defined by Equation 3.14 and 3.15. Fig. 3.4 summarizes what the values of these metrics mean for different ranges. Positive values indicate that the AS is benefiting from the locality policy. Negative values indicate that the AS is doing worse than before the policy is applied. For example, a profit increase larger than 0 means more



(a)



(b)

Figure 3.5. Individual AS deploys localization with Class pricing model. Sensitivity to locality models.

profit, while a profit increase between 0 and -1 means less profit. Profit goes to 0 when profit increase takes the values of -1. Finally, for values smaller than -1 the localization policy turns profit into loss. We show results in two different graphs: Fig. 3.4(a) plots the Cumulative Distribution Function (CDF) of loss reduction for ASes who have losses before localization, and Fig. 3.4(b) plots the CDF of profit increase for ASes who profit before localization.

In Fig. 3.4(a), the Lower bound (i.e., the vertical curve at x=0) shows that no profit is gained. This is because the localization of P2P traffic will result in internal

P2P clients reducing content downloaded from the outside, but in the pessimistic case this will not necessarily reduce content uploaded to other ISPs. However, for the worst case, the AS is charged on outbound traffic which has not changed. As Class reveals though, most residential ASes do get charged for their incoming traffic and thus localization is beneficial to them. Benefits are somewhat limited, with a loss reduction smaller than 30% for more than 50% of ASes. Also for Class, note that a few residential ASes are outgoing dominated and thus are unaffected by localization. In our dataset we find 40 ASes that belong to this category. For those, loss reduction is equal to 0, as shown by the vertical segment of the Class curve close to y=0.

For Average, less benefit is obtained than for Class because Average considers both directions of traffic but the cost associated with outbound traffic remains the same. Finally, Upper bound provides optimistic predictions that are unlikely in practice. Surprisingly even in this case the loss reduction is limited, i.e., only 40% of ASes see reductions over 60%.

We now turn our attention to profitable ASes in Fig. 3.4(b), a total of 16 residential ASes for which most P2P traffic traverses customer links. We see that in Class, 63% of these ASes show a profit reduction. This is due to these residential ASes also being transit ASes. For example, the AS that suffers the most is AS-209 Qwest Communications, a Tier-1 provider who we found to have over 360,000 clients. This is due to almost all of the P2P traffic that clients in AS-209 generate being sent and received through customer links.

Insight #2: *Some residential ASes will actually lose profit when they localize traffic. This is due to these ASes also being transit providers for other residential ASes. For these ASes, P2P traffic that was previously downloaded from clients in customer ASes decreases due to localization and in turn revenue decreases. Therefore, they have little incentive to localize traffic.*

There are a few ASes that are able to increase profit due to localization. This is due to the fact that many AS paths are asymmetric. Specifically, outgoing traffic is sent on customer links and since outgoing traffic does not decrease when one AS localizes, revenue remains the same. However, some incoming traffic is received on provider links, hence a reduction in costs and an increase in profit. This underlines the complexity of possible impacts of P2P traffic localization policies.

Sensitivity to locality model: Now we fix the pricing model to Class and vary the locality model. As expected, Fig. 3.5(a) shows that most ASes that were suffering losses due to the P2P traffic are reducing their loss due to localization policies. However, the reduction is not as large as one could hope. Under Perf(no history), the most pessimistic locality policy, for 75% of ASes the reduction is less than 25%. Even under Single(history), the most realistic locality policy, the loss reduction is still small, with less than 48% reduction for 75% of ASes. This is due to the small number of clients interested in the same content, which therefore tends to "disappear" as clients leave the torrent. Indeed, under the Single(persistent) policy the results are much improved: even 50% of ASes reduce their losses by 70%. In some cases, the AS is able to vastly improve profitability. For example, some small residential AS would be able to increase its loss reduction from 13% under Single(history) to 82% under Single(persistent). This is due to the optimistic assumption that content is available forever once it enters an AS.

Insight #3: *Content availability plays a crucial role in determining the effectiveness of localization. Due to churn, peers will often need to redownload content from outside the AS. However, when assuming persistent content, most ASes can reduce losses twice as much.*

3.5.3 Internet-wide Localization Deployed

We now consider the scenario when all ASes deploy localization at the same time and thus we also include ASes who are purely transit in our results. We show results on sensitivity to locality models, but not on results concerning sensitivity to pricing models as the trends are similar to those already seen.

As before, we fix the pricing model to Class and plot results separately for ASes that normally lose or profit due to P2P traffic. Fig. 3.6(a) plots the loss reduction and shows results similar to when individual ASes localize. This is because an AS will reduce its incoming traffic only if it localizes its own traffic. Thus, as most ASes are inbound dominated they can unilaterally localize traffic and receive the full benefits. However, an AS will reduce its outgoing traffic only if other ASes localize their traffic. Therefore, ASes that are outbound dominated will not see benefit until other ASes start localizing traffic. In this scenario indeed, all ASes that were facing loss reduce costs (loss reduction greater than 0 for all ASes).

Insight #4: *The benefits of localization will be limited for some ASes unless all ASes start to localize traffic. Localization, if adopted by a single AS, only reduces traffic received by internal peers, but it may not affect traffic sent. Hence, individual ASes that are outbound dominated or are charged based on the average of the inbound and outbound P95s will not receive all the possible benefits. This reduced benefit may slow down the adoption of localization policies.*

To investigate which ASes benefit more, we show in Fig. 3.6(b) the loss reduction versus population size, considering the Single(history) locality policy. As can be seen, there is a trend that the larger the population, the more the AS can localize. For example, the Taiwanese AS-3462 where we found over 1.5 million clients, is able to get a reduction of 91%. However, more than 50% of ASes achieve gains smaller than 30% as the limited number of peers interested in the same content inside an ISP limits the benefits of localization.

Insight #5: *The reduction in traffic due to localization only grows logarithmically with client population (notice the log-linear scale). Furthermore, we find that for all locality models the values of α_j, the leftover traffic, also follow a similar logarithmic trend with respect to AS population sizes. This is due to torrent popularity following a Zipf-distribution, which has been shown to limit the effectiveness of caching. In particular, [48] demonstrates through analysis that a similar effect occurs considering web caching.*

Moving to ASes that were already profitable, Fig. 3.6(c) shows a significant decrease in the amount of profit; in a pessimistic case – Single(no history) policy – 50% of ASes lose over 25% of their profits. In an optimistic case – Single(persistent) policy – 80% lose at least 60% in profit. Thus, while localization is beneficial for many residential ASes, over 300 transit ASes lose profit. Further investigation shows that the larger the transit AS is, the more likely it will suffer heavier losses in profit.

Insight #6: *Transit ASes lose significant amounts of profit when ASes localize. We found that all Tier-1 ISPs will lose over 56% of profits on P2P traffic under Single(history) when all ASes localize.*

Some ASes turn from being profitable to actually losing money (profit increase smaller than -1). For example, this happens for the AS-3786, who is a transit provider for the AS-17858. As AS-17858 has more peers than AS-3786, it can reduce its traffic more than AS-3786 can. Therefore, AS-3786's customer traffic is reduced more than its provider traffic and hence it starts to lose money. Interestingly, there are a few ASes that are able to increase their profits due to localization. These transit ASes are providers for many small residential ASes. As small ASes achieve very small reductions, the transit ASes are able to increase their profits by reducing their costs more than their customers can.

Insight #7: *Small residential ASes have small reductions in traffic due to the logarithmic trend of localization. Hence transit ASes who carry traffic for many small ASes fare better than those who carry traffic for a few large ASes.*

3.5.4 Localization Deployed by Large ASes

Besides the extreme cases when a single AS or all ASes deploy localization, we also investigate the scenario when ASes with larger populations will implement localization. We consider the Single(history) locality model and conduct sensitivity to pricing models. Results for sensitivity to locality models are similar.

We first consider when only the 100 largest ASes by client population size localize traffic, i.e., 6% of residential ASes in our dataset. As the largest ASes send and receive a very large amount of P2P traffic, we expect the localization to impact many other ASes as well. Fig. 3.7(a) shows the loss reduction results. In Class, the 100 ASes that localize receive the full benefits while 87% of ASes do not practically benefit. This is because many ASes are inbound dominated, but only outbound traffic decreased in this scenario. Indeed, the 40 ASes that are outbound dominated benefit with a loss reduction of 30% or greater. The Lower curve corroborates this result by showing that most ASes cannot get any benefit. The Average pricing model presents a "what-if" scenario that allows more ASes to benefit from the localization deployment of few ASes, while the Upper Bound provides over-optimistic an prediction.

Insight #8: *Pricing scheme has a large impact on the effectiveness of savings. As the maximum pricing model ignores one direction of traffic, reduction in the other direction does not result in a reduction of cost. The average pricing model does consider both inbound and outbound traffic and thus an AS could benefit both if it or some other AS localizes traffic.*

We now take the most realistic pricing model, Class, and to explore a "what-if" scenario we compare it with Average, when a varying number of ASes localize traffic. We only focus on loss reduction graphs as we wish to highlight the effects the maximum and average pricing models have on costs. Fig. 3.7(b) shows Class and demonstrates that those who localize are generally the only ones who see benefit. This is in contrast to the Average pricing model, which we show in Fig. 3.7(c). Interestingly, in Average, almost all ASes that do not localize see increasing benefits as more ASes localize. For example, when 200 ASes localize, most ASes have over a 20% loss reduction, which is over 50% of the benefits possible when all ASes localize.

Insight #9: *Contrary to the average pricing model, for the maximum pricing model it is not sufficient that few ASes localize traffic to reduce cost. Even if the largest Ases start deploying localization schemes, overall loss reduction will be very limited.*

3.6 Impact of Business-Relationship Policies

In this section we explore alternative ways to increase profitability of carrying P2P traffic. In particular, we explore business-relationship based peer selection policies where ASes aim to improve their profit by making internal peers select external peers located in customer or peer ASes, while trying to avoid peers hosted in provider ASes. Notice that the AS is not trying to reduce the amount of traffic peers will download, but rather it is interested in carefully selecting the source ASes to download from. Clearly, the generalized use of these policies could have significant impact on existing peering agreements. As traffic exchange ratios [49] are often used to determine if an AS should be a peer or customer, a change in traffic may lead to a renegotiation of agreements. In this chapter, we do not consider such events.

3.6.1 Modeling Business-Relationship Based Policies

To model this preferential peer selection, we define θ_{ij} as a preference bias index given by AS j to remote AS i. If the path from i to j traverses a customer link of j, the preference will be the highest ($\theta_{ij} = 1$); if the path from i to j traverses a peering link of j, a middle preference will be assigned ($\theta_{ij} = w_p, 0 < w_p \leq 1$); finally, if the path from i to j traverses a provider link of j, the preference will be the lowest ($\theta_{ij} = w_q, 0 < w_q \leq w_p$). Then, the volume of P2P traffic sent from AS i to AS j is:

$$X'_{ij} = X_{ij}\theta_{ij}B(j), \tag{3.16}$$

where X_{ij} is computed based on the Affinity model as in Equation 3.3, and $B(j)$ is a normalization factor that ensures the aggregate traffic downloaded by peers in AS j from external peers remains the same before and after the policy is applied.

$$B(j) = \frac{\sum_{i=1}^{D_j} X_{ij}}{\sum_{i=1}^{D_j}(X_{ij}\theta_{ij})}, \tag{3.17}$$

where D_j is the total number of ASes from which j downloads content. We refer to this model as the *Business* model.

We have performed a sensitivity study to w_p and w_q, to understand how these parameters affect the loss reduction and the profit increase of ASes. Intuitively, ISPs should make w_p and w_q as small as possible to obtain the most benefits out of Business. In the extreme, if we make $w_q = 0$, all the traffic from an AS will be directed to customer or peering links. However, in practice this may not be possible since customer or peer ASes of an ISP may not have the content or may not have enough clients to support the demand. Hence, we pick a very small value of w_q, in particular we use w_q =1E-10. For w_p, the main requirement is that it is larger than w_q; we choose w_p =1E-03. We note that Business is an extreme version of such a scheme that we use to illustrate its potential. In reality, other practical considerations should be made, such as considering user performance and inter-AS link capacities.

Intuition suggests we can improve the performance of the Business and Single policies by merging them. We call the new policy *Hybrid* and we model it by substituting L_{ij} from Equation 3.4 into both Equation 3.16 and 3.17 , i.e., $X'_{ij} = L_{ij}\theta_{ij}B(j)$. This represents the policy for selecting peers outside an AS to obtain content that is not already present inside the AS. We use w_p =1E-03 and w_q =1E-10 as before.

3.6.2 Best Strategy for Individual ASes

The goal of this section is to study what strategy individual ASes should adopt to have the best impact on ISP profitability. We start by considering the case in which individual ASes deploy one of Business, Hybrid or Single(history). We fix the pricing model to Class. Fig. 3.8(a) shows a comparison between the three possible strategies reporting loss reduction. Interestingly, the Business policy is ineffective for more than 75% of ASes, while Single(history) has proved to reduce the loss for most ASes. The Hybrid policy provides the best loss reduction for most of the ASes. Indeed, only for the top 25% of ASes, which are mostly transit ASes, Business performs better than Single(history) and similar to Hybrid. This is because transit ASes can benefit more from the Business policy by having internal peers download traffic from customer ASes rather than provider ASes. In fact, the top 11% of ASes actually turn profitable, i.e., loss reduction becomes greater than 1.

Fig. 3.8(b) shows the profit increase for the 16 residential ASes that are already profitable before localization. The figure shows that Business is the most beneficial policy, i.e., more than 30% of the ASes improve their profit by more than 100%. The other two policies can instead cause a profit reduction, as already seen in Fig. 3.5. Recall indeed that transit ASes will increase their profit if more traffic is pushed to customer ASes.

Based on these results, we aim to study the strategy that gives the most benefits to ASes. Towards this goal, we plot Fig. 3.8(c). In this figure, we consider all ASes and group them according to the categories described in Sec. 3.5. Then, we find for each AS, which policy gives the most benefits. Finally, we aggregate the best policies per category of AS. For each type of AS there is a stacked bar, which indicates the fraction of ASes that performs the best with a given policy. Note that besides the Business and Hybrid policies, there is $Single(history) = Hybrid$, which accounts for the cases in which both Single(history) and Hybrid are the best policies. Single(history) is never better than Business or Hybrid, so it is not shown in the picture.

There are several points to take away from Fig. 3.8(c). First, we observe that for around 90% of stub ASes, the best policy is Single(history) or Hybrid. This is because stub ASes receive considerable benefits from localization. ASes in the Small ISP category follow a similar trend with more than 60% of them benefiting the most from Single(history) or Hybrid. Second, all Tier-1 ASes on the contrary will get the most benefits out of the Business policy. This is because Tier-1 ASes will benefit from an increase in the traffic sent or received from customers. Finally, Hybrid is better for the Large ISP category, since these ISPs benefit both from directing traffic to customers and from localizing their own P2P traffic.

Insight #10: *Many ASes will achieve more profits through preferentially directing traffic to customers and peers rather than localizing traffic. Therefore, P2P traffic localization is not always the best choice for all ASes.*

3.6.3 Internet Impact of Business-Relationship Based Policies

In the previous section, we have seen how different strategies will benefit ASes if individual ASes adopt them. But what happens when all ASes adopt the same policy at the same time or when all ASes adopt their local best policy at the same time? Both P4P and ALTO indeed allow each AS to run their own "oracle" and chose a different policy. To answer these questions, we consider the scenarios in which all ASes adopt Business, Single(history), and Hybrid policies. In addition, we consider the scenario in which each AS applies its best local strategy, according to Fig. 3.8(c), which we have called "Individual Best". We fix the pricing model to Class.

Fig. 3.9(a) shows the loss reduction. For about 10% of the ASes, Business causes them to lose considerably more. These are mostly stub ASes that will be "victims" of their providers that increase the amount of traffic they exchange with customer ASes. On the contrary, Single(history) almost never causes higher loss. Hybrid performs marginally better than Single(history) for large ASes, but slightly worse than Single(history) for small ASes.

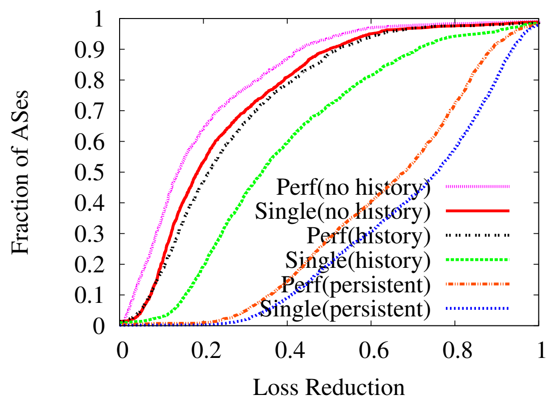
Fig. 3.9(b) shows the profit increase. Business performs better than Single(history) and Hybrid. When Business alone is considered, more than 70% of ASes either profit more or earn the same amount as before. For Single(history), over 90% of the ASes start losing profit due to localization. This is because many of the transit ASes that were profiting before will receive more benefit from Business since they will now select peers in customer ASes and direct more traffic to them.

We note that for both loss reduction and profit increase, Individual Best closely follows Single(history) and Hybrid. In particular, ASes that profit from P2P traffic (e.g. Tier-1 ASes and a few Large ISPs), which benefit more from locally implementing Business, lose because of policies implemented by their customers.

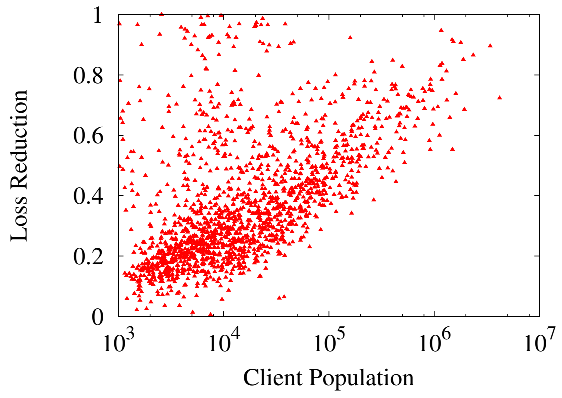
Insight #11: *While business-relationship based policies may locally be the best strategy for some ASes, they can have a negative external impact on other ASes. Furthermore, as the best local strategy of an individual AS is chosen in isolation of others it* *does not turn to be the best possible choice when all ASes deploy their own best local strategy.*

3.7 Summary

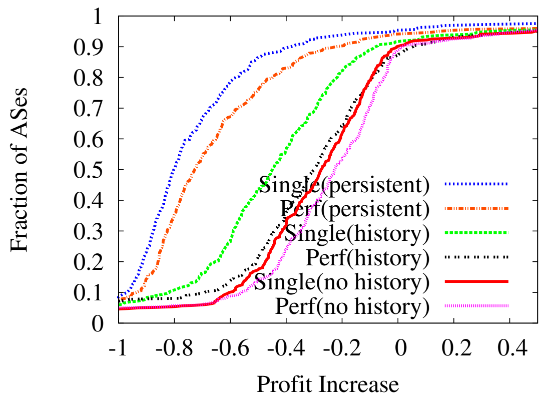
In this chapter, we developed a detailed methodology for evaluating the profitability of an ISP and how it will change due to P2P localization. We first proposed the Affinity model, a refinement of the Gravity model, for generating realistic inter-AS P2P traffic. We then devised several locality models to describe the reduction of P2P traffic under different scenarios. Coupling these models with realistic inter-AS paths inferred from BGP and IXP data, and pricing models based on the 95th percentile and geographic pricing, we calculate the impact of localization policies on ISP profits. We believe that the results we presented enhance the understanding and implications of P2P traffic localization schemes on the Internet, and in particular from the perspective of ISPs.



(a)

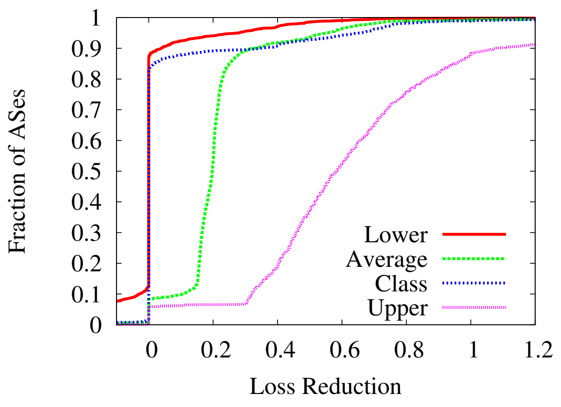


(b) Impact of client population on profit.

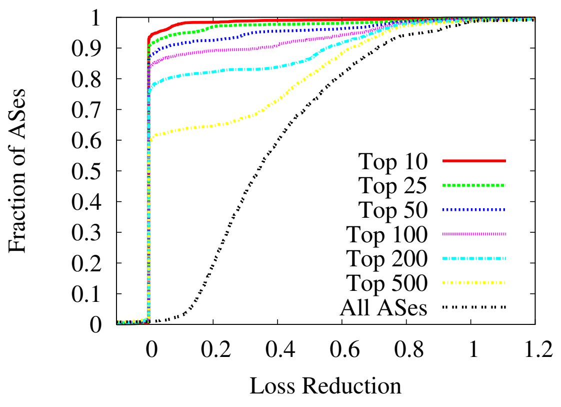


(c)

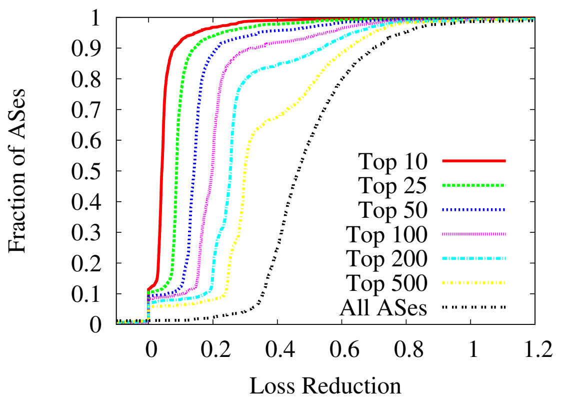
Figure 3.6. All ASes deploy localization with Class pricing model. Sensitivity to locality models.



(a) Largest 100 localize.

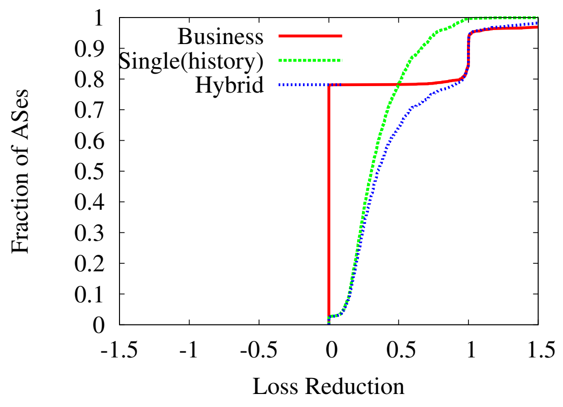


(b) Class pricing model.

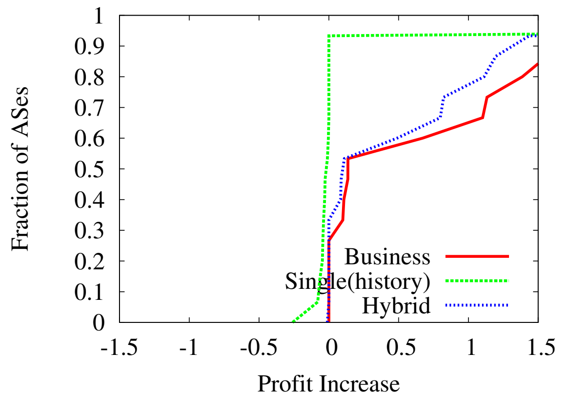


(c) Average pricing model.

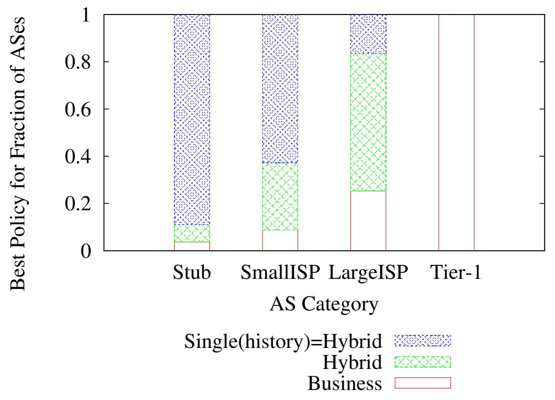
Figure 3.7. Largest ASes deploy localization with Single(history) locality model. Sensitivity to pricing models.



(a)

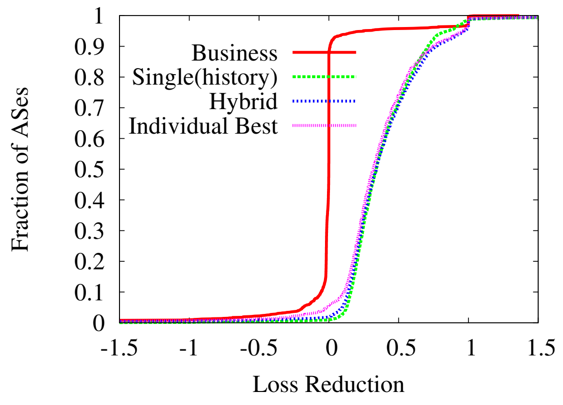


(b)

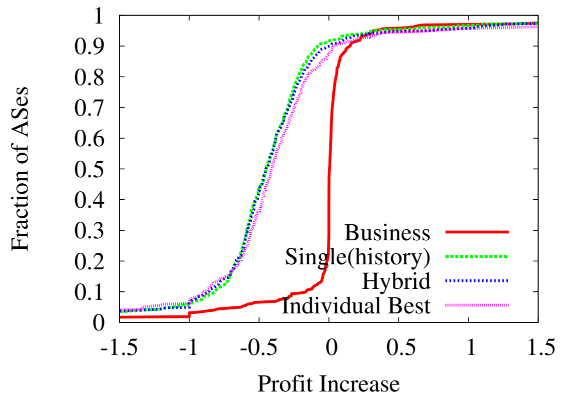


(c)

Figure 3.8. Individual AS deploys Business, Single(history) or Hybrid, with Class pricing model.



(a)



(b)

Figure 3.9. All ASes deploy Business, Single(history), Hybrid or Individual Best, with Class pricing model.

4 SECURING VIRTUAL COORDINATES BY ENFORCING PHYSICAL LAWS

As discussed in Chapter 3, localization services provide network locality to distributed systems, causing the amount of traffic that must leave an ISP to decrease. The reduced amount of traffic brings benefit to some ISPs as their costs can also decrease. Not only can ISPs benefit from localization, but also numerous distributed protocols that can take advantage of network locality, such as optimized replica placement [50], multicast tree and mesh construction [51], routing on the Internet [52,53], and Byzantine fault-tolerant membership management [54]. Given how critical such services are, in this chapter we study how to protect Virtual Coordinate Systems (VCS) from insider attacks aiming to disrupt them.

4.1 Introduction

Virtual Coordinate Systems (VCS) have been proposed as an efficient and low cost service to provide network locality estimations by accurately predicting round-trip times (RTT) between arbitrary nodes in a network. Each node measures the RTT to a small number of other nodes and the VCS then assigns a coordinate to each node. Each node can then estimate the RTT between itself and any arbitrary node by calculating some distance function.

While some VCS are centralized in nature [55], many have been designed as distributed systems [56], where each node maintains and updates its own coordinate by relying on information received from other nodes. Distributed VCS can be classified as landmark-based and decentralized. Landmark-based systems [57–61] assume a trusted set of nodes that form the infrastructure by which other nodes can determine their coordinates. Decentralized VCS [56, 62–64] assume no such infrastructure; a node updates its coordinate based on measurements and information from a random set of nodes.

Unfortunately, distributed VCSs have been shown [65] to be vulnerable to insider attacks, where compromised nodes delay measurement probes and lie about their coordinates to decrease system performance. As many applications rely on VCS to build robust services, there have been several proposals to secure them. For example, outlier detection [66, 67] and voting [68] were used to detect equivocation of lying attackers. Most of these defense methods ultimately decide if an update from a node is malicious or not by learning good behavior through system observation over time. As a result, these schemes are vulnerable to attacks where through small changes attackers make the defense mechanisms learn malicious behavior as being good behavior. One such attack is the well-known frog-boiling attack where attackers lie by small amounts that accumulate over time and gradually lead to large changes in performance [69–71].

A classical approach for designing distributed systems is to use *safety invariants* in order to ensure system correctness. These safety invariants specify states into which the distributed system should never enter. For example, a distributed system that forms a tree of nodes should never have any loops, or a distributed hash table should never form multiple rings, but only one continuous ring. At first glance, VCS do not appear to have such invariants as minimal constraints are imposed on how neighbors are selected or on what coordinates a node can possibly have. We make the key observation that some VCS are designed around an abstraction of a physical system [55, 56, 72] and that physical systems follow physical laws. As these laws are universally true, we can leverage them to identify safety invariants for VCSs based on physical systems.

In this chapter we present Newton, a decentralized VCS which extends Vivaldi [56] to withstand a wide class of insider attacks by using safety invariants derived from Newton's three laws of motion. Newton relies on the observation that Vivaldi is an abstraction of a real-life physical system and therefore all participating nodes must follow Newton's three laws of motion. As there is a direct mapping between the actions taken by nodes, in reporting their coordinates and RTTs, and the forces that these physical laws govern, any attack in which malicious nodes lie about their coordinates or delay probes will result in the invariants being violated. We leverage this fact to detect attacks and discard malicious updates. Our contributions are:

- We describe how to use Newton's three laws of motion as well as a mapping between forces and virtual coordinates to identify invariants that mitigate a wide range of attacks against Vivaldi. We show how to use the three identified invariants to detect and mitigate the well-studied inflation, deflation, and oscillation attacks, as well as the more recent frog-boiling and network-partition attacks.

- We conduct extensive simulations and real-world experiments on PlanetLab to demonstrate that Newton is able to mitigate all five attacks mentioned above.We compare Newton with Vivaldi outfitted with Outlier Detection [67] and show that Newton is not vulnerable to the frog-boiling and network-partition attacks. We also find that, even with no attackers, Newton has better performance than Vivaldi, i.e. Newton is 25% more accurate and 68% more stable.

- We consider extreme scenarios where the attackers are present in a much higher percentage, over 50% of nodes in the network are malicious, and also where attackers are conducting attacks from the beginning of the experiment, while the system has not converged yet to a steady state. We show that even under such conditions Newton still performs well.

- We consider adaptive attackers that know how the invariants are used and try to exploit them. Because in real-deployments Newton is not a perfect abstraction of a physical system, an attacker can try to exploit the invariants. We explore a new type of attack, *rotation attack*, where attackers rotate their positions slowly around the origin of the coordinate plane in an attempt to destabilize nodes while remaining undetected. We find that Newton holds up well to such attacks, incurring only slightly worse accuracy.

The remainder of this chapter is organized as follows: We describe Vivaldi in Sec. 4.2 and attacks against it in Sec. 5.3. We describe Newton and our invariants in Sec. 4.4. We show simulation results in Sec. 4.5 and PlanetLab experimental results in Sec. 5.7. We present our summary in Sec. 4.7.

4.2 Vivaldi Coordinate System

Algorithm 1: Node i Coordinate Update

```
Input: Remote node tuple $\langle x_j, e_j, RTT_{ij} \rangle$
Output: Updated local coordinate and error $x_i$, $e_i$
1 $w = e_i/(e_i + e_j)$
2 $e_s = |\|x_i - x_j\| - RTT_{ij}|/RTT_{ij}$
3 $\alpha = c_e \times w$
4 $e_i = (\alpha \times e_s) + ((1 - \alpha) \times e_i)$
5 $\delta = c_c \times w$
6 $x_i = x_i + \delta \times (RTT_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$
```

Vivaldi [56] is a decentralized VCS where the distance between coordinates represents the estimated RTT between nodes. All nodes start at the origin and periodically update their coordinates based on interaction with a subset of nodes referred to as the neighbor set. A node chooses half of these nodes randomly from all possible nodes and the other half from a set of low-latency nodes. Research [56] has shown that a neighbor set of 64 nodes ensures quick convergence.

In addition to the coordinate value, each node also maintains a local error value which shows the confidence in the coordinate. Algorithm 1 describes how each node i updates its coordinate. Specifically, i will send a request to node j for its coordinate and local error value. When node j replies node i also measures the actual RTT.

An observation confidence w is calculated first (line 1) along with the error e_s in comparing the coordinates with the actual RTT (line 2). Node i updates its local error (line 4) by calculating an exponentially-weighted moving average with weight α and system parameter c_e (line 3). Next, i computes the movement dampening factor calculated with another system parameter c_c (line 5) and updates its coordinate by finding how far it should move and then multiplying that by a unit vector (represented by $u(\bullet)$) in the direction it should move (line 6).

A VCS generally has the system goals of providing accuracy and stability with respect to the coordinates that it produces. Accuracy describes how closely the coordinates reflect the actual RTT between nodes. Stability describes how quickly nodes converge to a set of accurate coordinates and how long a node can be absent from the system and still have accurate coordinates.

Accuracy. We use *prediction error* to measure accuracy: $Error_{pred} = |RTT_{Act} - RTT_{Est}|$, where RTT_{Act} is the measured RTT and RTT_{Est} is the estimated RTT. A small prediction error indicates high accuracy. We report the median of all the prediction errors at a time instant.

Stability. We use *velocity* of a node to measure stability: $Velocity = \frac{\Delta x_i}{t}$, where Δx_i is the change in coordinates for node i (or distance traveled by a node), and t is the amount of time taken to make that change. A small velocity indicates high stability. We report the average of velocity of all nodes at a time instant.

4.3 Attacks Against VCS

We consider that a bounded number of compromised and colluding nodes act maliciously. To attack Vivaldi, a malicious node can (1) influence the coordinate value computation by lying about its coordinate and local error value or (2) influence the RTT computation by delaying the measurement probe.

An attacker can exploit coordinate and RTT computation to conduct the following *basic attacks*:

• **Inflation:** Attackers lie about having very large coordinates. This pulls benign nodes far away from their correct coordinates and thus is an attack on accuracy.

• **Deflation:** Attackers lie about having small coordinates near the origin. This prevents benign nodes from being able to update to their correct coordinates and therefore is also an attack on accuracy.

• **Oscillation:** Attackers lie by reporting randomly chosen coordinates and randomly delaying measurement probes. This is an attack both on accuracy and stability.

Basic attacks against Vivaldi have been shown to be very effective in reducing accuracy and stability [65]. Moreover, while defenses have been proposed, recent work [69–71] identified more *advanced attacks* that are able to bypass all previously proposed defenses [66–68]. Advanced attacks are:

• **Frog-boiling:** Attackers lie by small amounts at a time, slowly increasing this amount by moving their coordinates in one direction. Over time though, the attacker ends up reporting coordinates that are far away from their correct coordinate. This results in an attack on both accuracy and stability.

• **Network-partition:** Attackers lie similarly as in the frog-boiling attack, but instead groups of nodes collude together and move in opposite directions, again attacking both accuracy and stability.

4.4 Description of Newton

In this section we present our VCS, Newton, which builds upon Vivaldi by implementing invariants derived from physical laws to defend against all known insider attacks against VCS.

4.4.1 Vivaldi as a Physical System

The coordinate update in Vivaldi is actually modeled based on a mass-spring system abstraction, where each pair of nodes have a spring connecting them. Depending on its state, the spring applies a force to the nodes to either push them together or pull them apart. This force is calculated by Hooke's law, $F = -kx$, where k is a spring constant and x is the amount of displacement that a spring currently is from its equilibrium or rest position. Every node has a spring constant k value of 1. To determine displacement, the measured RTT between a pair of nodes is considered to be the length of the spring at its rest position, while the current length of the spring is the estimated RTT. Over time, the system stabilizes when all pairs of nodes minimize the amount of force that is placed upon them.

When updating its coordinate based on information from node j, a node i calculates the magnitude and direction of the force $\vec{f_{ij}}$ that node j is applying to it. The magnitude of the force m_{ij} is determined by the RTT between the two nodes and the distance of the current nodes' coordinates: $m_{ij} = RTT_{ij} - \|x_i - x_j\|$. The direction of the force $\vec{d_{ij}}$ is a unit vector that is calculated based on the two nodes' coordinates: $\vec{d_{ij}} = u(x_i - x_j)$. The force $\vec{f_{ij}}$ is then simply $\vec{f_{ij}} = m_{ij} * \vec{d_{ij}}$. This determines how much the coordinate needs to be updated from the previous value and corresponds to Line 6 in Algorithm 1. Note that Vivaldi is not a perfect physical system and also takes into account the perceived error reported by the node j and its own local error value. We discuss the implications of Vivaldi not being a perfect physical system in Sec. 4.4.5.

4.4.2 Using Physical Laws to Identify Invariants

Detecting insider attacks in distributed systems can benefit from identifying invariants in the system. For Vivaldi, no such invariants appear to exist at first glance since nodes make decisions based on inputs from nodes in their neighbor set and there are no constraints imposed by the system in node selection. We make the key observation that since Vivaldi [56] is built upon an abstraction of a mass-spring system, all nodes must follow physical laws. These laws are universal truths so they represent invariants that all nodes in Vivaldi should globally follow. In particular, nodes must follow Newton's three laws of motion which are:

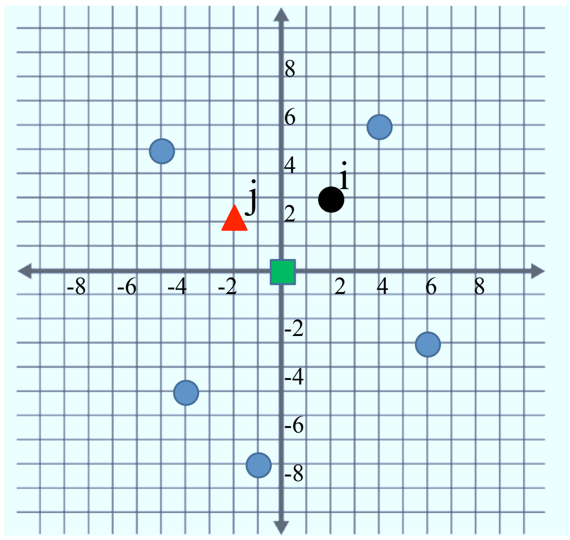
First law: *A body stays at rest unless acted upon by an external, unbalanced force.*
Second law: *A force F on a body of mass m undergoes an acceleration a, such that the acceleration is proportional to the force and indirectly proportional to the mass.*
Third law: *When a first body exerts a force on a second body, the second body exerts an equal but opposite force on the first body.*

When an attacker lies about its own coordinate, it is implicitly lying about forces that have previously acted upon it, thus introducing extraneous *indirect forces* into the system. Introducing such forces into the system breaks the first and third laws, as attackers are not acting according to the influences of the outside forces upon them. When an attacker delays a measurement probe or lies about its local error value, it is lying about the force between itself and another node, thus introducing extraneous *direct forces* into the system. Lying about such forces breaks the second law, as nodes do not undergo accelerations that are governed by the forces determined by Hooke's law.

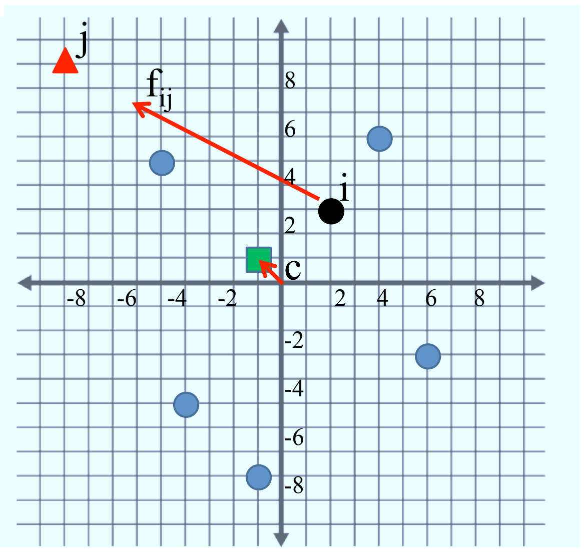
We show how to leverage Newton's three laws of motion to identify three invariants, which we call **IN1**, **IN2** and **IN3**. Nodes can then use these invariants to *locally* detect whether an update that results in a force being acted upon is the result of nodes behaving according to the protocol and thus following physical laws, or the result of a lying attacker. Below we define the invariants and describe how to detect extraneous indirect and direct forces with their help.

4.4.3 Detecting Extraneous Indirect Forces

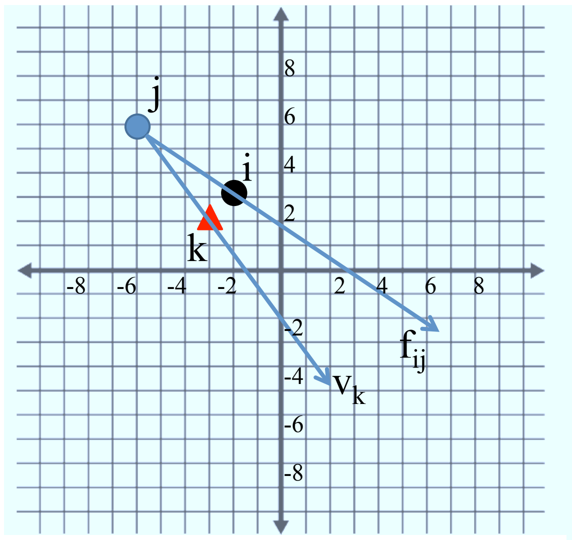
We first focus on how to detect whether a node is lying about the forces that have acted upon it, resulting in maliciously derived coordinates. For ease of exposition, assume each node i is at coordinate x_i and at any moment is applying the force $\vec{f_{ij}}$ onto node j. As described in Sec. 4.2, a node chooses its neighbor set based on two criteria: (1) half are chosen randomly and (2) half are chosen based on if they are



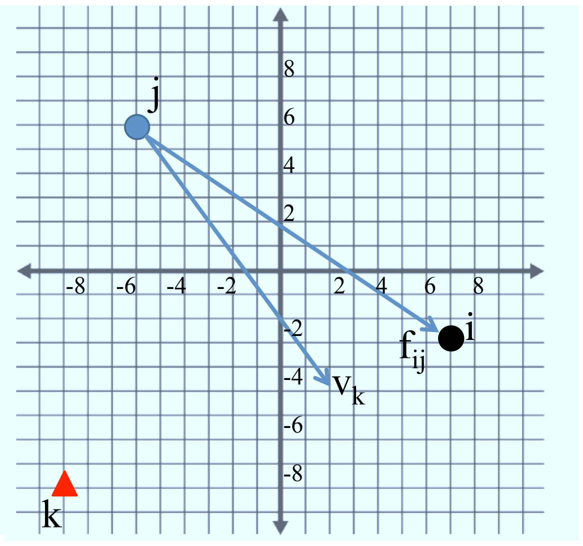
(a) Random: no attack



(b) Random: attack



(c) Physical close: no attack



(d) Physical close: attack

Figure 4.1. Detecting extraneous indirect forces

physically close. We design two detection schemes, one for nodes that are randomly chosen, and the other for nodes that are physically close.

Detection for malicious random nodes from the neighbor set: We observe that the third law states that there can be no unbalanced forces in the mass-spring system. An attacker introducing any extraneous indirect force that causes nodes to move will be an unbalanced force by definition of the first law. The third law then implies that an unbalanced force can be detected by finding the centroid of all the node's coordinates, where the centroid is the average of all the coordinates and has the physical analogue of being the center of mass of the mass-spring system. We note that while perfect detection requires knowledge of the coordinates of all nodes, using just the randomly selected nodes also provides a good vantage point from which to calculate an approximate centroid. We summarize our first invariant.

IN1: *If the centroid of a node i and the randomly selected nodes from its neighbor set is at the origin then no unbalanced force has been introduced. However, if the centroid is not at the origin, then an attacker (or collection of attackers), has introduced an unbalanced force that has the same direction as a force vector from the origin to the centroid ($\vec{c}$).*

In Figs. 4.1(a) and 4.1(b) we illustrate how to use **IN1** to detect attacks. In Fig. 4.1(a) node i, located at coordinate x_i = (2,3), is the victim and all the other dots are the randomly selected nodes from its neighbor set, including node j. Node i can calculate the centroid c based on its own coordinate and the coordinates of all those neighbors $c = \frac{\sum_{p=1}^{n} x_p}{n}$. Since the third law states that all forces must be balanced, we would expect that the centroid would never move, and thus even during normal operations, would be at the origin. In Fig. 4.1(a) the green square signifies this calculated centroid, and since no attack has taken place yet, it is at the origin.

In Fig. 4.1(b), we consider what happens when the attacker, node j, represented by the red triangle at coordinate x_j = (-2,2), introduces an extraneous unbalanced force. In this case, the attacker moves to coordinate (-9,9). Node i recalculates the centroid, using $c_t = \frac{\sum_{p=1}^{n} x_{p_t} + f_{ij}}{n}$, to be at coordinate (-1,1), which corresponds to $\vec{c}$, the force that moved the centroid from the origin. Node i also experiences a force $\vec{f_{ij}}$, represented by the arrow pushing it towards the attacker. Node i can detect the attack by finding that $\vec{c}$ is non-zero, as described in **IN1**. It can then find which node introduced the unbalanced force, and thus is the attacker. Specifically, for every neighbor node k, i sums up the forces ($\vec{s}_{ik}$) that k has applied to it since k entered its neighbor set and then calculates the vector projection of $\vec{s}_{ik}$ onto $\vec{c}$. The node whose projection has the greatest magnitude is the one who has contributed most to the centroid being moved, thus an attacker, and its force is ignored.

IN1 holds even if a malicious node initially reports an incorrect coordinate because the system always starts in a correct state (all the nodes start at the origin, and so does the centroid).

Detection for malicious physically close nodes from neighbor set: For nodes that are physically close, we observe that because all nodes are connected via springs they will experience very similar forces from the same nodes. We can use the first law, which dictates that a node in a mass-spring system must move if acted upon by an external, unbalanced force. Moreover, the second law implies that we can detect if a node should be moving or not and we can calculate how much it should move. Our second invariant can now be summarized:

IN2: *Nodes i and k are physically close and if node i experiences a force $\vec{f_{ij}}$ from node j, then node i would expect node k to experience a force from j similar to the vector projection of $\vec{f_{ij}}$ onto the vector $u(x_j - x_k)$.*

We use Figs. 4.1(c) and 4.1(d) to illustrate **IN2**. Fig. 4.1(c) shows the nodes before the attack. The black dot at coordinate (-2,3) is node i, the victim, the blue dot at coordinate (-6,6) is node j, and the red triangle at coordinate (-3,2) is the attacker node k. Both i and k experience forces upon them from j. Node i can calculate what it expects the force upon k to be and thus determine that it expects k to update its coordinate to (2,-5).

Fig 4.1(d) shows the nodes when the attack happens. Node i does move according to the force applied to it to coordinate (7,-3). However, when k attacks by introducing an extraneous indirect force, it moves in a different direction than expected. To detect the attack, node i can calculate the force value for node k as described in **IN2** for every force that is applied to itself and sum up that value ($\vec{v}_k$). Node i will remember the previous coordinate that was reported by k and when it receives a new update from k it calculates the change (Δx_k). This difference and the sum of vector projections $\vec{v}_k$

should be equivalent, if they are not, then k did not move according to the external unbalanced force.

4.4.4 Detecting Extraneous Direct Forces

We now focus on how to detect if a force directly acting on a node is extraneous and is caused by a malicious process. To accomplish this, we leverage the second law of motion and Hooke's law. The second law states how much a node should accelerate given the force and mass of a node. In our mass-spring system, the mass of every node is 1, and thus can be ignored. In a mass-spring system, the amount of force applied to a node is controlled by Hooke's law, $F = -kx$ which states that the amount of force on a node is proportional to the spring's current displacement from its rest position. We now state our third and final invariant:

IN3: *As the springs in the physical system stabilize and come closer to their rest position, nodes should decelerate and thus the forces that are applied to them should decrease over time.*

IN3 applies also to joining and leaving nodes. While joining nodes may lie about their initial force, **IN3** obliges a decreasing force over time. Leaving nodes stop moving and the force becomes zero.

One possible detection scheme is to impose a certain rate of decrease on the forces applied to a node, and if the force is larger than expected, offending nodes are considered malicious. However, we have experimentally found that this approach is too strict for real deployments, due to practical aspects of the Internet. First, triangle inequality violations result in nodes stabilizing even though springs are still exerting force on nodes. Thus we can expect forces to never decrease all the way to zero, but rather opposing forces will simply be balanced. Second, **IN3** assumes that latencies do not change as real springs can not change their rest position. However, on the real Internet this will not hold as routes change and mobile nodes move.

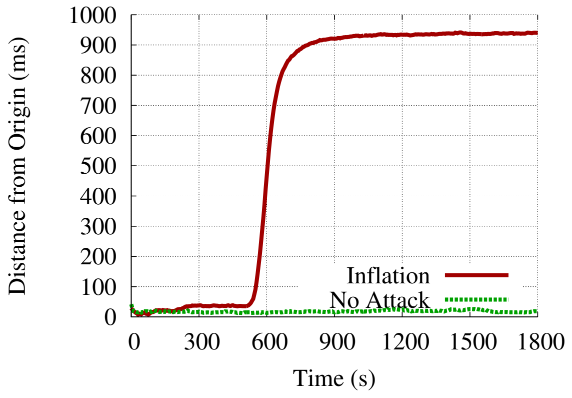
We instead take a different approach. A node calculates the median $\tilde{f}$ and median absolute deviation D of the magnitude of the force that each node is applying to it. Then if the magnitude of any force m_j is a few deviations larger than the median $m_j > \tilde{f} + k * D$, the node will ignore it. We use the median and median absolute deviation instead of the average and standard deviation, as the former are more robust to outliers and have been shown to be resilient against frog-boiling attacks [73].

4.4.5 Using IN1, IN2, and IN3 to Design Newton

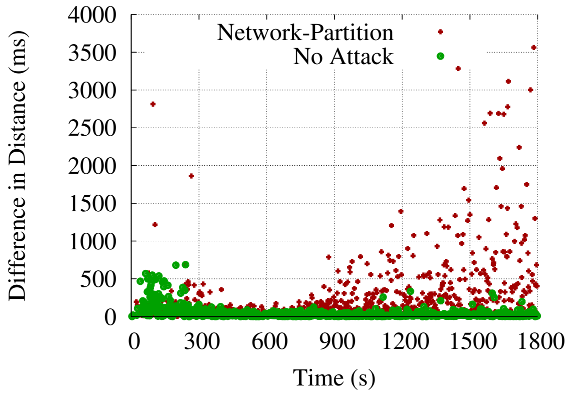
We use **IN1**, **IN2**, and **IN3** combined with Vivaldi to create Newton. We investigate if these invariants hold on real deployments of Vivaldi on PlanetLab. While Vivaldi models a mass-spring system, the actual protocol, and more importantly, any network on which it runs, will not perfectly emulate a physical mass-spring system. Thus, we expect some discrepancy between the ideal physical system and the real deployed system. We investigate these discrepancies and use the results to calibrate Newton.

We use results of Vivaldi on PlanetLab deployments of 500 nodes to investigate the invariants. We implement all 5 attacks (inflation, deflation, oscillation, frog-boiling, and network-partition) and plot results relevant to each invariant. As inflation and deflation share similar characteristics, with inflation being a more damaging attack, and network-partition is a stronger variant of the frog-boiling attack, Fig. 4.2 shows results for inflation, oscillation, and network-partition. Each attack starts at 600 seconds into the experiment and are conducted where 10% of nodes are attackers.

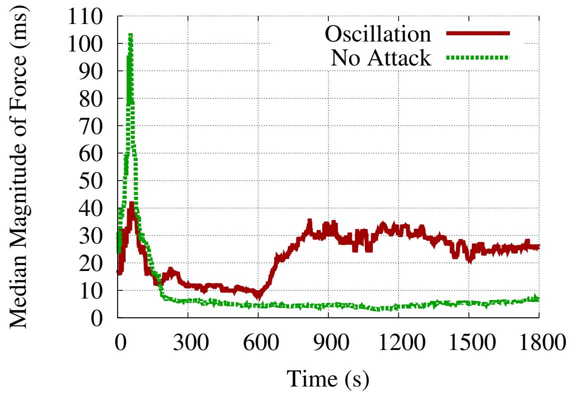
IN1. In Fig. 4.2(a), we plot the distance from the origin to the centroid of the coordinates of randomly chosen neighbor nodes, averaged for all nodes in the system. We expect this distance to be zero or very small. When there is no attack, we find the centroid to be less than 20 ms away from the origin. However, during an inflation attack, the value increases drastically as nodes start to lie about their coordinate. We select a threshold of 20 ms to detect an attack.



(a) **IN1** with Inflation



(b) **IN2** with Network-Partition



(c) **IN3** with Oscillation

Figure 4.2. Invariants shown through deployments of Vivaldi on PlanetLab.

Table 4.1
Sensitivity on threshold for IN2

Threshold (ms)	FPR	TPR
10	0.57	0.98
15	0.37	0.97
20	0.27	0.95
25	0.19	0.91
30	0.14	0.90
35	0.11	0.84
40	0.09	0.83
45	0.08	0.75
50	0.06	0.61

IN2. In Fig. 4.2(b) we plot the difference in distance from where physically close nodes were expected to have their coordinates located at, versus where they actually reported themselves to be. We expect this value to be zero or very small. When there is no attack on Vivaldi, we find these values to be small, with most less than 50 ms. When under a network-partition attack, these values increase dramatically, especially the further a node has moved from its correct coordinate. To find a good value for the threshold we conducted a sensitivity study by varying it between 10 and 50 and then finding the true positive rate (TPR) and the false positive rate (FPR) when classifying updates. We show the results in Table 4.1 and found a good threshold for detecting the attack to be 35 ms, which trades-off discarding some benign updates for better detection of malicious nodes.

IN3. Fig. 4.2(c) depicts the median of the magnitude of the force applied to a single node over time. We see that while there is not a strictly decreasing line as one would expect in an ideal system, the general trend is present. Also, in an oscillation attack we see that this value quickly grows and is inconsistent with benign behavior.

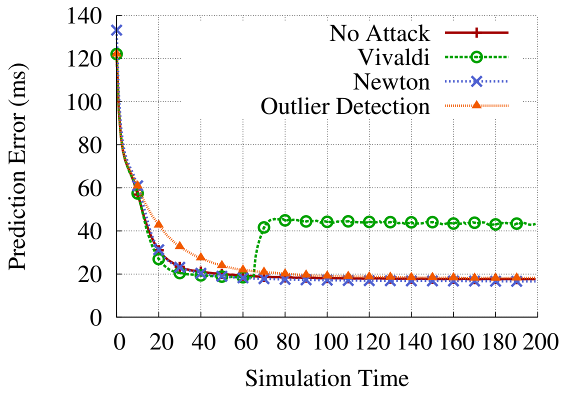
To detect attackers, we have found that it is best to calculate the median separately for randomly chosen nodes and physically close nodes. This is due to physically close nodes having smaller force values, but deviate more from the median, while randomly chosen nodes have the opposite characteristics. Thus we choose a threshold of 8 absolute deviations for physically close nodes and 5 for randomly chosen nodes.

Implementation. To implement Newton, we started with the base code of Vivaldi and then added the invariants. In Newton, every node checks the invariants after receiving an update from another node. If at least one invariant is violated the update is discarded.

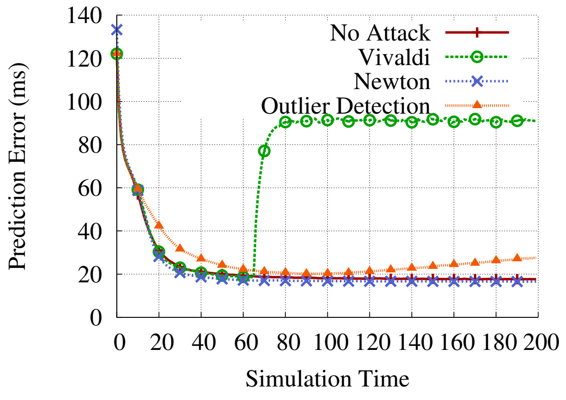
Thresholds. Because Newton uses thresholds that rely on a fixed point as a reference, such as the origin, they are more difficult to exploit by an attacker. Nevertheless, an attacker can still try to exploit these thresholds by staying under their values.We discuss scenarios where an attacker can exploit these thresholds in Sec. 4.5.4 and show that Newton is robust even under such scenarios.

Overhead. As Vivaldi is an efficient and low cost service for latency estimation, we also aimed to preserve that goal in designing Newton. As such, we do not add any extra network communication, as the use of our invariants do not require it, and the added computation and memory usage are very small.

Non-Euclidiean spaces. Since Newton is based on physical laws found in our Euclidean-based world, we investigate if Newton works in non-Euclidean spaces. We show results for Newton in hyperbolic spaces in Sec. 4.5.3. Furthermore, as the most general form of non-Euclidean spaces are Riemannian manifolds, and as the Nash embedding theorem says that any m dimensional Riemannian manifold can be embedded isometrically in some Euclidean space, we see that the defined invariants would still hold in non-Euclidean spaces. However, the construction of this isometrical embedding is not straightforward and if pseudo-Riemannian manifolds are used for virtual coordinates then no such embedding might exist.



(a) Accuracy – 10% attackers

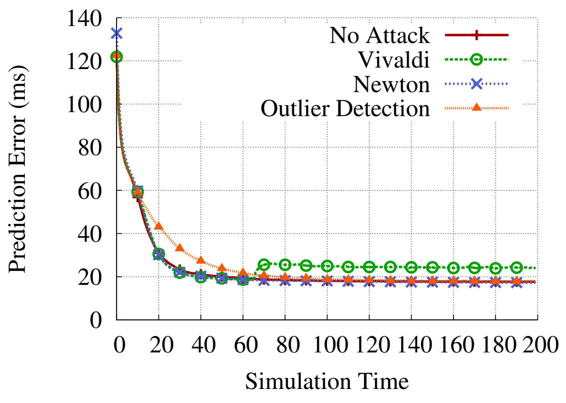


(b) Accuracy – 30% attackers

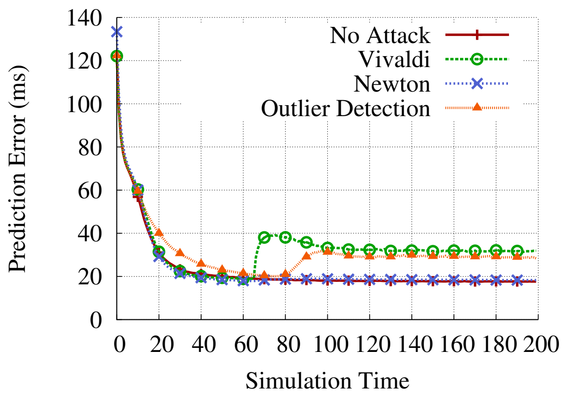
Figure 4.3. Simulation results – inflation attack

We show through simulations, using the p2psim simulator [74], how effective Newton is in defending against attacks. We compare Newton against the unsecured Vivaldi and also Vivaldi outfitted with Outlier Detection [67], referred to as *Outlier Detection*. We also include Vivaldi when no attackers are present, referred to as *No Attack*, as a baseline comparison.

We use the King data set [75] which contains Internet pairwise measurements between 1740 nodes (average RTT is 180 ms and maximum RTT is 800 ms). Simula-



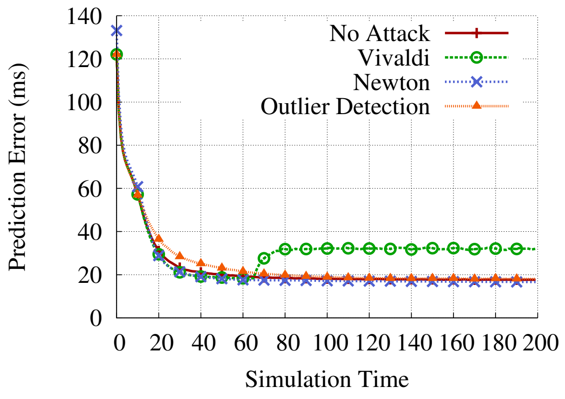
(a) Accuracy – 10% attackers



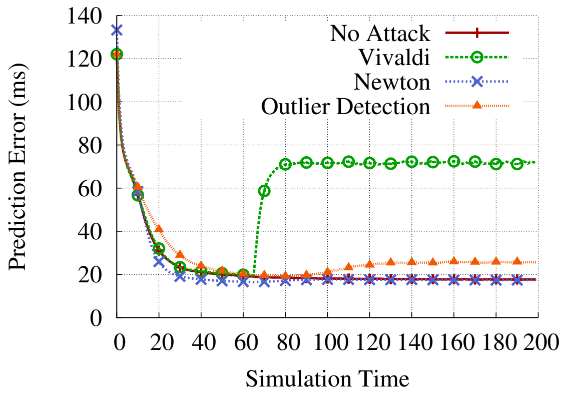
(b) Accuracy – 30% attackers

Figure 4.4. Simulation results – deflation attack

tions last for 200 time units, where each time unit is 500 seconds. Each node joins at the beginning of the simulation in a flash-crowd scenario and remains for the entire duration. We use a typical setting for Vivaldi [56], where every node has a neighbor set of 64 nodes, with half randomly chosen and the other half being nodes with low RTT (also referred to as physically close nodes). The attackers are chosen randomly from all nodes. Unless otherwise stated, malicious nodes start their attack at one-third of the way through the simulation. This is to give a fair comparison for Outlier Detection, as it needs to learn what good behavior is. Outlier Detection uses spatial and temporal thresholds of 1.25 and 4, respectively, as described in [67]. Newton



(a) Accuracy – 10% attackers



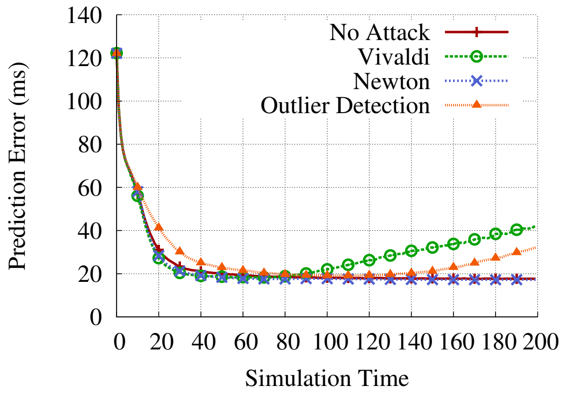
(b) Accuracy – 30% attackers

Figure 4.5. Simulation results – oscillation attack

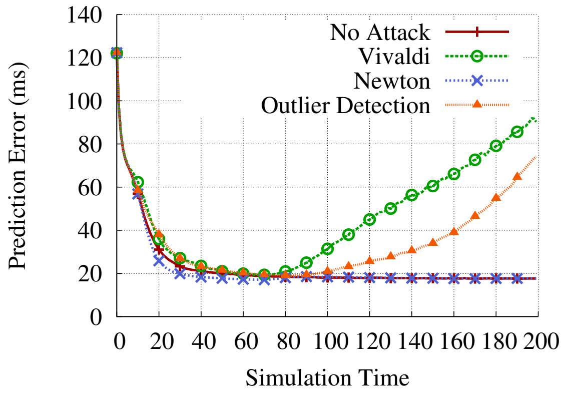
uses the thresholds described in Sec. 4.4.5 which any Internet-wide deployment could use. For the coordinate space, we use a Euclidean distance and gradient function in 2 dimensions, unless otherwise stated.

4.5.1 Attacks Mitigation

We vary the percentage of nodes that are attackers between 10%, 20%, and 30%. Due to similarity of results we show only the 10% and 30% cases.



(a) Accuracy – 10% attackers



(b) Accuracy – 30% attackers

Figure 4.6. Simulation results – frog-boiling attack

Inflation. Figs. 4.3(a) and 4.3(b) show the accuracy under an inflation attack. We can see that when under attack Vivaldi has very poor accuracy, which gets increasingly worse with the percentage of attackers. Both Outlier Detection and Newton are able to effectively keep the error low after the attack starts. However, as the percentage of malicious attackers increase, Outlier Detection's prediction error also increases as time progresses, while Newton is able to match the baseline prediction error. We attribute Newton's performance to its ability to detect that the attacker nodes are introducing unbalanced forces and thus shifting the centroid far away from the origin.

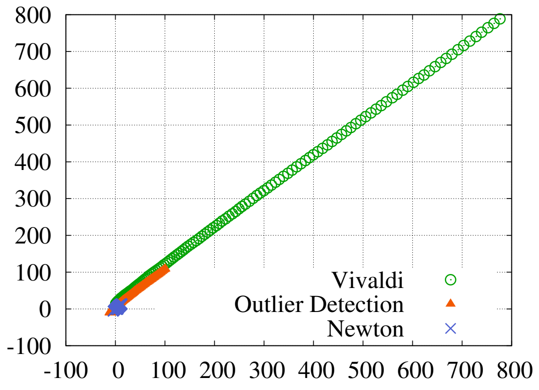
Deflation. Results for the impact of the deflation attack on accuracy are in Figs. 4.4(a) and 4.4(b). The deflation attack does not have as great of an impact on Vivaldi as inflation, but the opposite is true of Outlier Detection. However, we see again that Newton is able to successfully mitigate the attack.

Oscillation. The oscillation attack is different from the previous two attacks in that while attackers lie about their coordinates in a random way, they also delay measurement probes up to 1 second. We show the results of how the different systems handle the attack and the impact on accuracy in Figs. 4.5(a) and 4.5(b). Outlier Detection is able to withstand the attacks until there are 30% attackers, when the prediction error increases to 26 ms. However, Newton continues to provide good performance for all percentage of attackers. We attribute this to **IN3**, requiring forces to decrease over time.

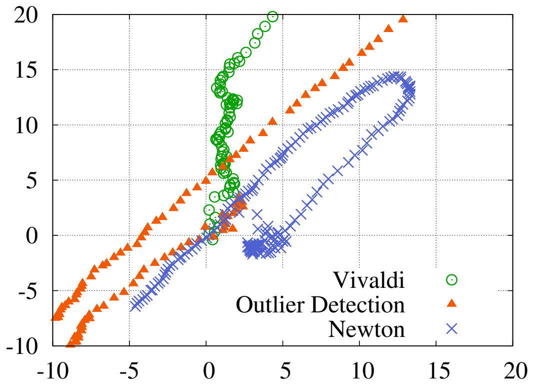
Frog-boiling. The frog-boiling attack, has been shown in [69–71] to be an effective attack against VCS defenses that must learn over time what good behavior is. We now show the impact of the attack on accuracy in Figs. 4.6(a) and 4.6(b). Similar to previous works, we see that Outlier Detection indeed does not protect against such an attack. Newton, though, is able to successfully protect against the frog-boiling attack.

We give insights about how Newton works in Fig. 4.7(a) showing how the centroid moves over time on the coordinate plane when under attack (10% attackers). Vivaldi's centroid moves far away from the origin. Outlier Detection's centroid does not move as far, but still it moves close to (100,100). To be able to see how Newton's centroid moves, we show a zoomed in picture in Fig. 4.7(b). Newton's centroid also initially moves away from the origin, until it almost reaches coordinate (13,15). At this point individual nodes calculate that the centroid is near 20 ms away from the origin, thus triggering the detection mechanism. The honest nodes can then determine who the attackers are and ignore their updates.

Network-partition. The network-partition attack is similar to the frog-boiling attack, except multiple groups of attackers move in opposite directions, trying to split



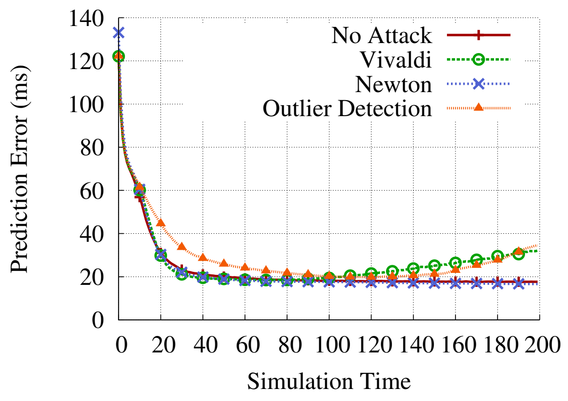
(a) Centroid



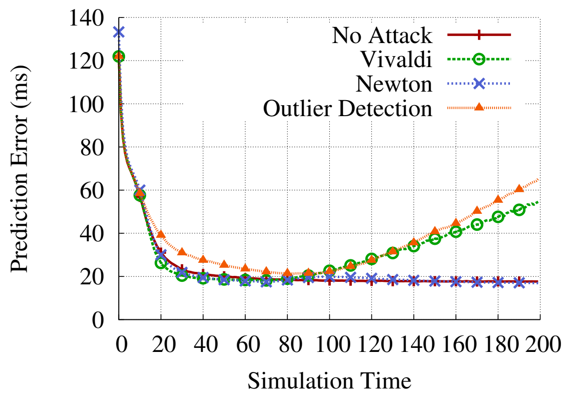
(b) Centroid of Newton

Figure 4.7. Centroid over time for frog-boiling attack

the network. We consider four groups of nodes moving in four different directions. Figs. 4.8(a) and 4.8(b) show the accuracy for the different systems under attack. This attack is successful against Outlier Detection, while Newton still performs well under attack. This is even though groups of attackers moving in different directions give the illusion that they are actually acting according to balanced forces by not moving the centroid, thus making it difficult to detect this attack using **IN1**. However, in this case, attackers that are physically close can still be detected by **IN2** and all types of attackers can be detected by **IN3**.



(a) Accuracy – 10% attackers

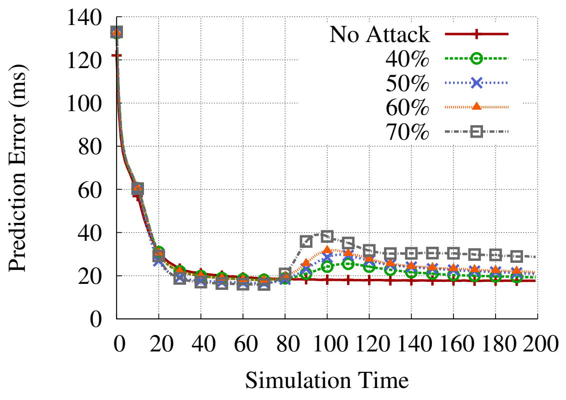


(b) Accuracy – 30% attackers

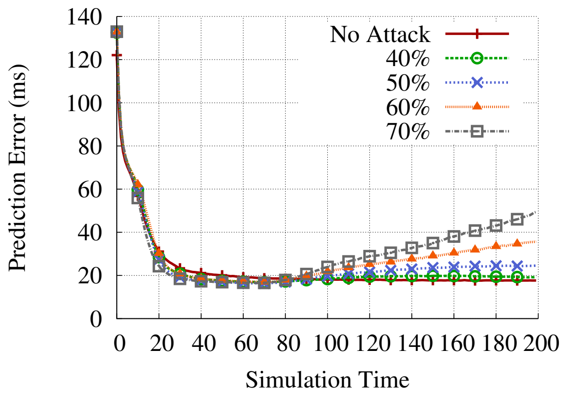
Figure 4.8. Simulation results – network-partition attack

4.5.2 Extreme Attack Scenarios

High percentage of attackers. We also show extreme scenarios where Newton must face an increasing percentage of attackers. We show the advanced attacks in Fig. 4.9, results were similar for the basic attacks, but we did not include them due to space constraints. Overall, we see that Newton is able to handle 50% attackers without losing significant accuracy. However, under 60% and 70% of attackers accuracy starts to degrade, particularly for the network-partition attacks. We point out that each node updates its coordinate based on a set of 64 nodes, thus the high-percentage



(a) Frog-boiling

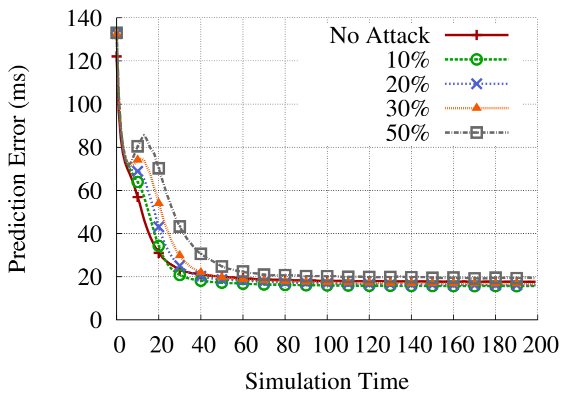


(b) Network-partition

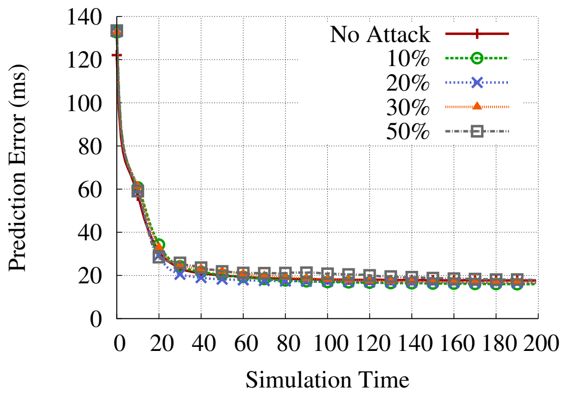
Figure 4.9. Simulation results – high percentage of attackers

of malicious nodes results into a lower percentage in the neighbor set. For example, using the analysis from [67], when there are 70% malicious nodes in the entire network, about 54% of nodes will be malicious in the neighbor set and thus able to manipulate the median that is used to detect extraneous direct forces.

Attacks before system converges to steady state. In previous simulations, we showed performance when there was a period before attacks started to allow Outlier Detection to learn good behavior. Newton does not need such period since it is based on invariants. We show results only for oscillation and frog-boiling attacks as results for the other attacks were similar. Fig. 4.10 shows results when attacks



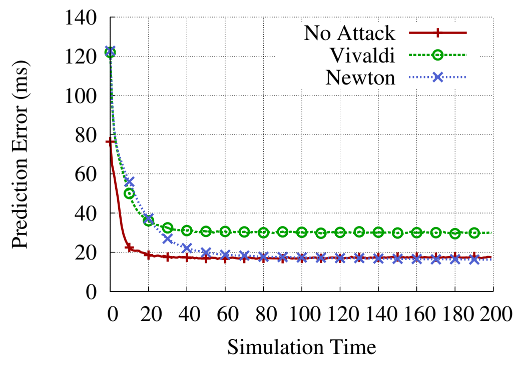
(a) Oscillation



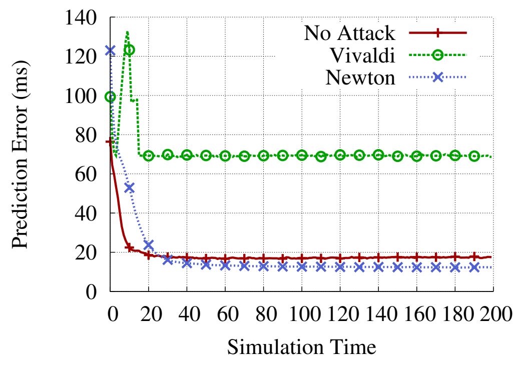
(b) Frog-boiling

Figure 4.10. Simulation results – attacks start at the beginning

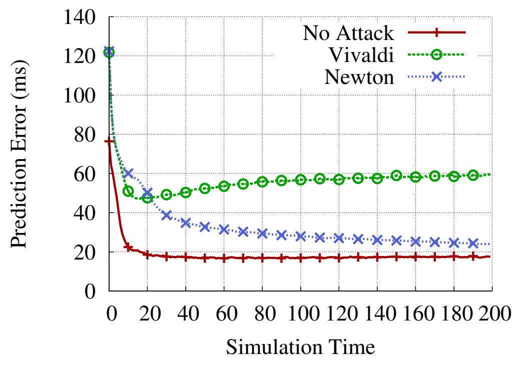
start from the beginning of the simulation. As can be seen, Newton mitigates the attacks. Under the oscillation attack, as the percentage of attackers increase, it takes slightly longer for coordinates to stabilize and become accurate. This is because we do not enforce a strict rate of decrease on the amount of force between two nodes and instead use the median force to detect nodes. Nodes must first sample a number of forces before they can calculate the correct median. Thus, in Newton an honest node cannot immediately detect if a node is artificially increasing the force between itself and another node.



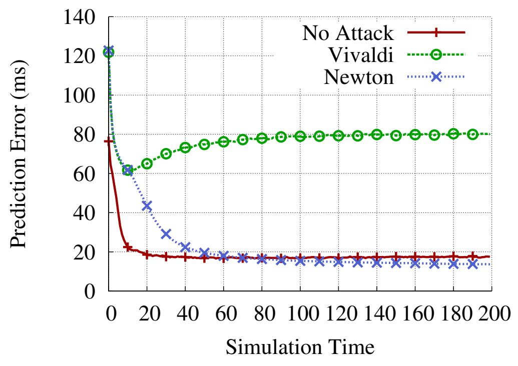
(a) Deflation attack



(b) Oscillation attack



(c) Frog-boiling attack



(d) Network-partition attack

Figure 4.11. Simulation results – accuracy when using 4 dimensions in hyperbolic space with 30% attackers

4.5.3 Newton in Higher-dimensional and Hyperbolic Space

So far we have shown that Newton works well in simple 2-dimensional Euclidian coordinate spaces. However, more complex spaces have been shown in the past to improve prediction error. For example, Ledlie *et al.* [76] have shown through a Principal Component Analysis that 4 dimensions are appropriate for Internet-scale network coordinates. Hyperbolic spaces also have been proposed as an alternative to Euclidean spaces as they better represent the structure of the Internet [77]. Several works have applied Vivaldi to such spaces and have shown that it does produce an accurate embedding [78, 79]. Modifying Vivaldi and Newton to work in hyperbolic spaces simply involves changing the distance and gradient function. We implement these functions as described in [79]. Hyperbolic spaces also have a curvature parameter that describes how much a line deviates from being flat. We experimentally found that a value of 60 provides good accuracy in benign environments. We ran simulations in hyperbolic space in 4 dimensions. We find that for 10% and 20% attackers, Newton performs better than the baseline. Newton continues to work well even under 30% attackers, which we show in Fig. 4.11.

4.5.4 Invariants under Attack

Because in real-deployments Newton does not behave exactly like a physical system, it uses thresholds for the three invariants. We note that Newton's thresholds use as a reference a fixed point such as the origin, while Outlier Detection's thresholds use as a reference a moving point (the centroid of metrics derived from all nodes in the neighbor set), allowing attacks such as frog-boiling to move it. Thus, Newton's thresholds are more difficult to exploit by an attacker. However, an adaptive attacker can still exploit the values of the thresholds used by Newton to his advantage.

We conduct three tests, one for each invariant, where the attacker tries to remain undetected, yet come as close to the threshold as possible. For **IN1**, the attackers push the centroid to right below the 20 ms threshold. For **IN2**, attackers initially move as the forces dictate, but then always shift just below 35 ms away from this position. Finally, for **IN3**, attackers delay probes only enough to stay beneath the deviation threshold. The results of these tests are shown in Fig. 4.12, where we zoom in on the results of the steady state performance to see the effects. We compare the normal baseline of Vivaldi when no attack occurs, Newton when no attack occurs, labeled *Newton 0%*, and also Newton when there are 30% attackers, labeled *Newton 30%*. We find that even 30% attackers can not significantly increase the prediction error.

Attackers can also conduct a new attack, which we call the *rotation attack*, where the goal is not necessarily to disrupt accuracy, but rather stability. In this attack, colluding nodes rotate around the origin in the same direction at a slow rate. This attack will not trigger **IN1**, and if done slowly enough, will bypass the thresholds of **IN2** and **IN3**. We implement this attack and show the results in Fig. 4.13 (notice the zoomed-in y axis scale). We find the accuracy in Fig. 4.13(a) to only be slightly raised over our baseline. Stability, as shown in Fig. 4.13(b), is also raised over Newton's normal levels, but is not yet worse than the baseline.

4.6 Experimental Results

We evaluate Newton in real-life experiments on the PlanetLab testbed. We use 500 nodes and run each experiment for 30 minutes, unless otherwise stated. Every second a node chooses one of its neighbors to probe and gets their coordinate update. We use Newton configured for a Euclidean coordinate space. Due to PlanetLab being an Internet-scale testbed, we use 4 dimensions as suggested by Ledlie *et al.* [76]. Malicious nodes start performing attacks immediately once the experiment starts. All other parameters are the same as in the simulations. We compare Newton with Vivaldi under attacks and consider as baseline Vivaldi with no attacks.

4.6.1 Performance in Benign Networks

We first show the results when there are no attackers in Fig. 4.14. Accuracy is shown in Fig. 4.14(a) where the prediction error is lower in both Vivaldi and Newton for PlanetLab than the simulations. This is most likely due to the smaller number of nodes involved as the error needs to be minimized for a fewer number of nodes. Furthermore, Newton only has a resulting prediction error of 9 ms, while Vivaldi has one of 12 ms. The difference in stability has also increased over the simulations, as shown in Fig. 4.14(b). Vivaldi has a resulting velocity of 0.8 ms/s, while Newton is only 0.25 ms/s. This increase in accuracy and stability is due to Newton being less sensitive to probes that get delayed occasionally as the result of benign occurrences such as queueing delays on routers.

Adapting to changes in the network. In real deployments, such as on PlanetLab, route changes will take place, potentially having an effect on **IN3**. To show that Newton can withstand such changes, we run Newton for *four days* on 350 nodes on PlanetLab. For this particular experiment we reduce the frequency of how often a node sends a probe to a neighbor to 5 seconds, all other parameters remained the same as before. We performed traceroutes between all-pairs of nodes before and after the experiment to estimate the number of routes changed. We conservatively only count routes as changed if they contain different routers and also have a difference in RTTs greater than 10 ms. We find that 12% of all routes changed.

Fig. 4.15 shows the results. Initially, Newton is able to stabilize within an hour to 6 ms of error. We attribute this smaller error, compared to the 9 ms seen earlier, to the smaller number of nodes that must embed coordinates. However, over time, Newton reduces the error even further to 3 ms. We also investigate in more details what happens when routes change. We find that in many cases the resulting change is not so large that **IN3** is violated. However, there are cases in which **IN3** is violated for a short period of time, for one of the two nodes. This is due to when a single path between routers change, it often affects many end-to-end routes for one node, thus causing RTTs to multiple neighbors to change simultaneously. Thus, one node will realize that many neighbors are putting extra force on it, and change its coordinate accordingly.

4.6.2 Attack Mitigation

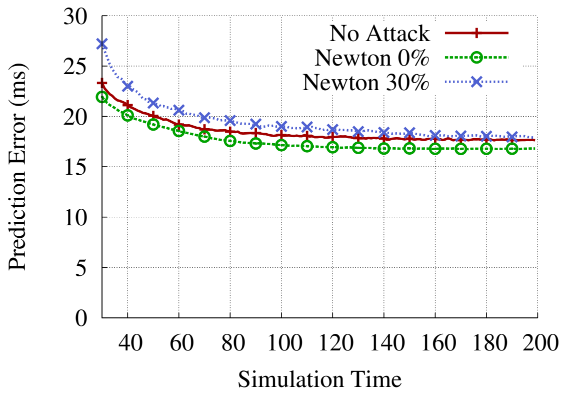
Inflation and deflation. Figs. 4.16 and 4.17 show accuracy under inflation and deflation attacks respectively, for 10% and 30% attackers in the system. We find that the inflation attack is not as effective against Vivaldi in these experiments as it is in the simulations, even though in the experiments we increased the amount that attackers lie about so that they have larger coordinates. The deflation attack is also not as effective as in simulations. In both cases, Newton is able to handle such attacks while having better accuracy than in the benign setting.

Oscillation. For the rest of the attacks, we show just 30% attackers. We conducted experiments with lower percentages of attackers, but we did not include them because of similarity of results. Fig. 4.18 shows accuracy and stability under the oscillation attack. This attack proves to be more damaging in the experiments than the simulations for Vivaldi. Newton though, because it is taking advantage of **IN3**, is able to mitigate such attacks easily.

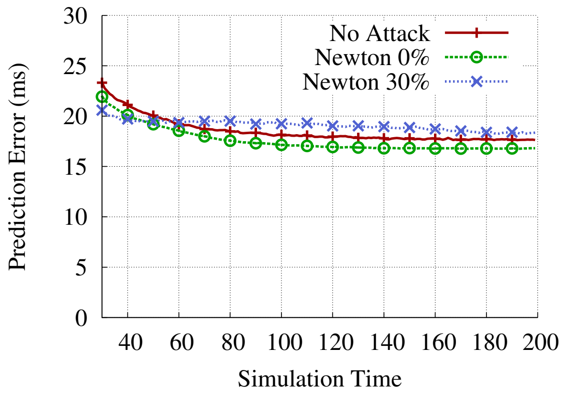
Frog-boiling and network-partition. Results for frog-boiling are shown in Fig. 4.19, which while we find it to be the most effective attack on Vivaldi, for reasons previously explained, it has no effect on Newton. Unsurprisingly, we find that the network-partition attack, which is similar to the frog-boiling attack but nodes move in different (four in our case) directions, has similar results to it. We plot the effects of this attack in Fig. 4.20.

4.7 Summary

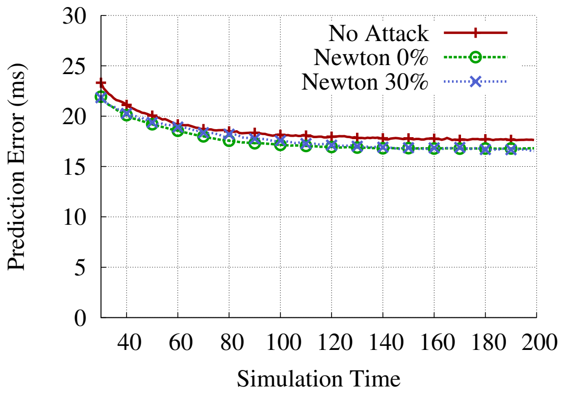
We introduced Newton, a new approach to providing a secure VCS by going back to the abstraction that Vivaldi is based on, a physical mass-spring system. In accordance with the abstraction, our defenses are based on the three laws of motion as put forward by Newton.We have explained in depth how the laws provide invariants for our system and how they are leveraged to mitigate basic attacks such as inflation, deflation, and oscillation but also more advanced attacks like frog-boiling, and network-partition attacks. Through simulations and experiments on the PlanetLab testbed we showed that Newton outperforms Vivaldi even in benign settings and is able to mitigate the advanced attacks that remained undetected by Outlier Detection. Newton can also cope with advanced attackers that might leverage insider knowledge about calibration specific parameters used by Newton. Newton is immune to such attacks, since the calibration of the defense mechanism is relying on robust and fixed, time independent thresholds.



(a) Attackers push the **IN1** threshold

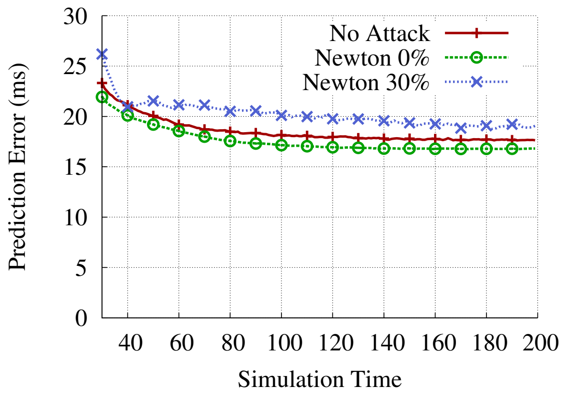


(b) Attackers push the **IN2** threshold

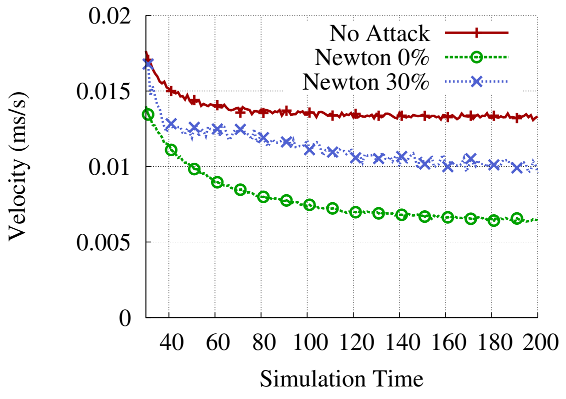


(c) Attackers push the **IN3** threshold

Figure 4.12. Simulation results – attackers (30%) push the limits of the thresholds used by the three invariants

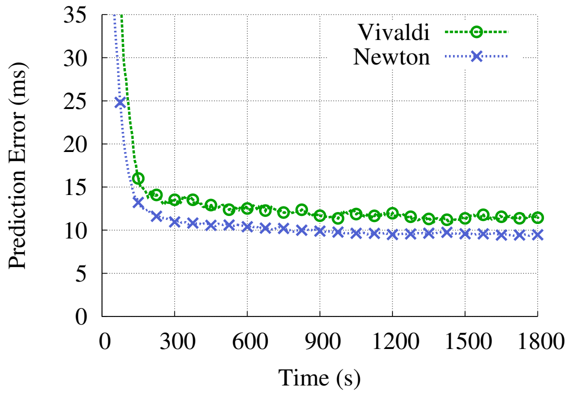


(a) Accuracy

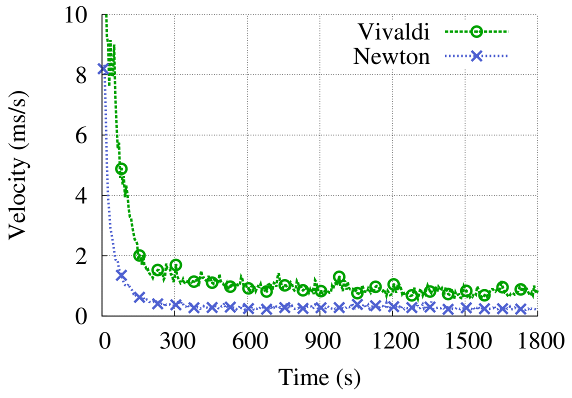


(b) Stability

Figure 4.13. Simulation results – attackers (30%) rotate around the origin at a slow rate



(a) Accuracy



(b) Stability

Figure 4.14. PlanetLab results – no attackers

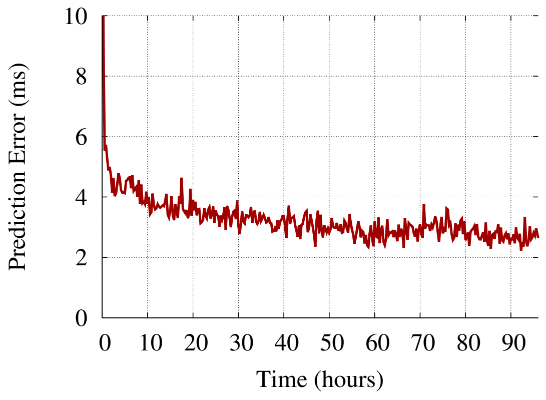
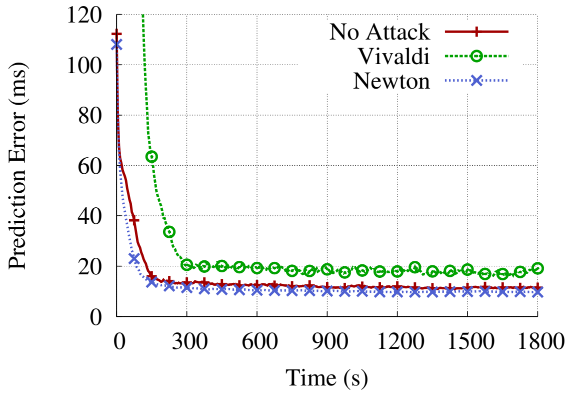
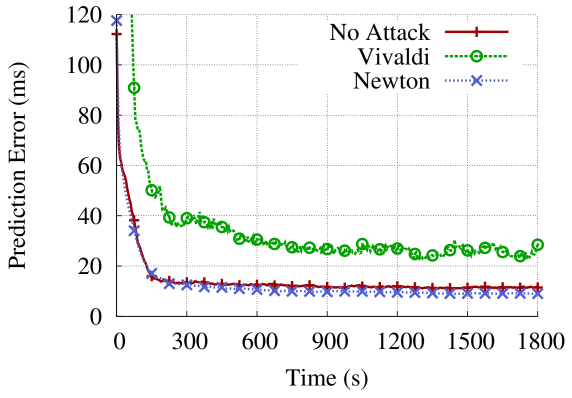


Figure 4.15. PlanetLab results – accuracy of Newton for 4 days

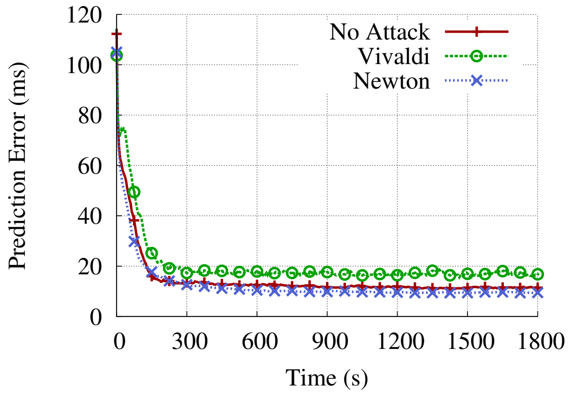


(a) Accuracy – 10% attackers

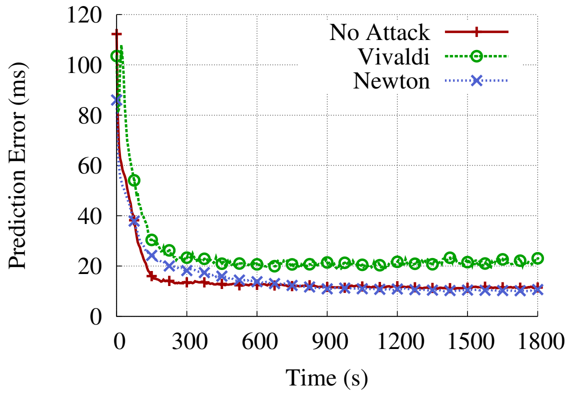


(b) Accuracy – 30% attackers

Figure 4.16. PlanetLab results – inflation attack

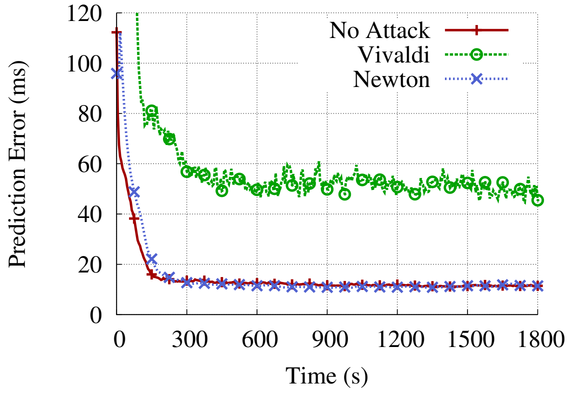


(a) Accuracy – 10% attackers

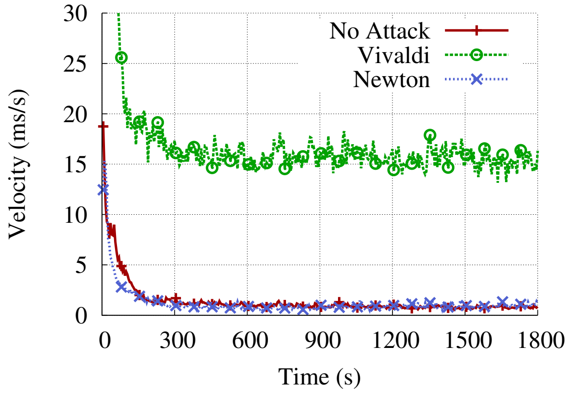


(b) Accuracy – 30% attackers

Figure 4.17. PlanetLab results – deflation attack

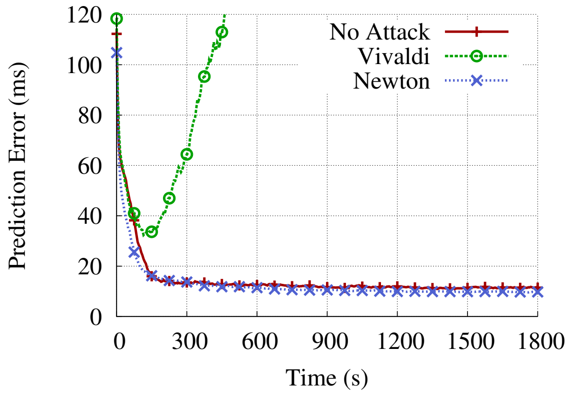


(a) Accuracy – 10% attackers

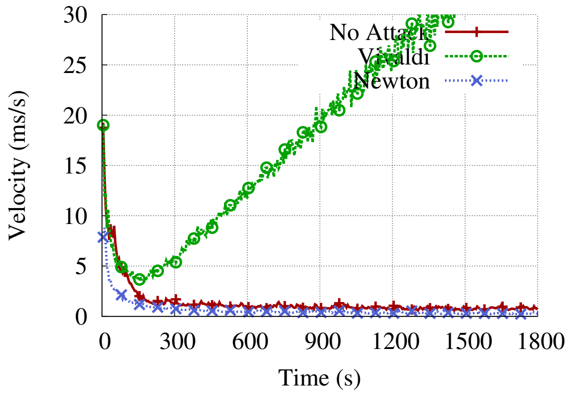


(b) Stability – 30% attackers

Figure 4.18. PlanetLab results – oscillation attack

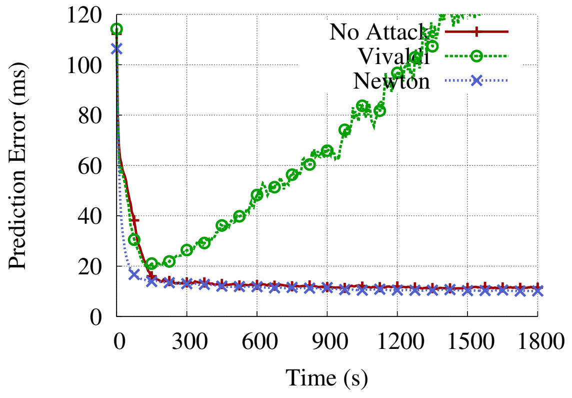


(a) Accuracy – 10% attackers

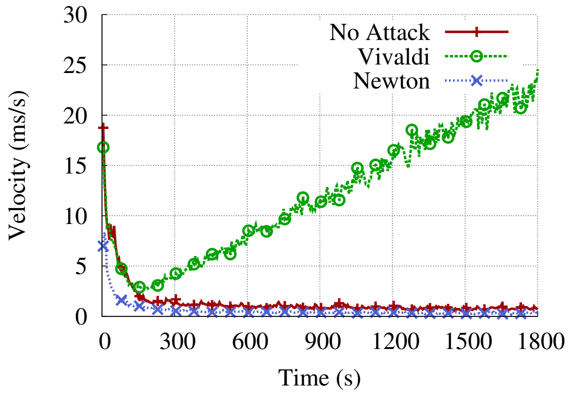


(b) Stability – 30% attackers

Figure 4.19. PlanetLab results – frog-boiling attack



(a) Accuracy – 10% attackers



(b) Stability – 30% attackers

Figure 4.20. PlanetLab results – network-partition attack

5 A DESIGN FOR SECURING DATA DELIVERY IN MESH-BASED PEER-TO-PEER STREAMING

As noted in Chapter 4, Virtual Coordinate Systems provide a localization service for overlays, however insider attacks can reduce the effectiveness of such services greatly. Similarly, insider attackers can target the overlays that run on top of localization services. In this chapter we discuss how to secure high-bandwidth P2P streaming systems against malicious insiders.

5.1 Introduction

The vision of enabling simultaneous video broadcast as a common Internet utility in a manner that any publisher can broadcast content to any set of receivers has been driving the research agenda in the networking community for over two decades. For much of the 1990's, the research and industrial community investigated support for such applications using the IP Multicast architecture [80]. However, serious concerns regarding its scaling, support for higher level functionality, and deployment have dogged IP Multicast. The sparse deployment of IP Multicast, and the high cost of bandwidth required for server-based solutions or Content Delivery Networks (CDNs) are two main factors that have limited broadcast to only a subset of Internet content publishers. While many network service providers have enabled IPTV services that deliver quality video to their own subscribers using packet switching, there remains a need for cost-effective, ubiquitous support for Internet-wide video broadcast.

Over the last decade, there has been significant interest in the use of peer-to-peer (P2P) technologies for Internet video broadcast [7, 10, 51, 81–83]. There are two key drivers making the approach attractive. First, such technology does not require support from Internet routers and network infrastructure, and consequently is extremely cost-effective and easy to deploy. Second, in such a technology, a participant that tunes into a broadcast is not only downloading a video stream, but also uploading it to other participants watching the program. Consequently, such an approach has the potential to scale with group size, as greater demand also generates more resources.

The extensive research in the design of P2P streaming systems [15, 51, 81, 84–87] has matured to the extent that we are today seeing several efforts aimed at commercializing the technology [7,10–12,88–94]. High user demand for these systems has been shown by their increasingly large user base [82, 83].

P2P streaming can be divided into two main approaches, tree-based [15,85,95,96] and mesh-based [84,86,97,98] architectures (see [99] for a survey). Tree-based overlays construct a tree, rooted at the source, which broadcasts the stream. Mesh-based overlays disseminate data in a less structured manner, where nodes exchange data with a subset of the nodes in the network without using any predefined route. Mesh-based approaches have received a lot of attention in recent times because they are more resilient to churn [100] and node failures, and have been shown to perform better than tree-based approaches [14, 100].

While mesh-based approaches have several attractive properties, the performance of these systems in the presence of malicious participants has received little attention. Dhungel et al. [101] show the vulnerability of such systems to attacks where malicious nodes upload polluted data to other nodes in the overlay. Similarly Haridasan et al. [102] focus on polluted data but also denial of service attacks on nodes by flooding them with requests. Several works have focused on the problem of peers which download data from other nodes but do not in turn upload data [103–106], however these works focus on selfish rather than malicious node behavior.

In this chapter, we systematically analyze the vulnerabilities of the components of mesh-based streaming overlays. We focus on an important and broad class of attacks where malicious nodes deliberately become neighbors of a very large number of nodes in the system and do not upload data to them. We also focus on attacks that are particular to streaming systems such as when malicious nodes artificially delay the uploading of data, so while nodes still receive the data, because of real-time deadlines they are less likely to have opportunities to forward that data to others. We focus on these attacks given they have received limited attention, they can have significant disruption on data delivery, and they are applicable to many mesh-based systems. For instance, our evaluation with a state-of-the-art mesh-based streaming system shows that when the attacks are conducted with just 10% of nodes in the system being malicious, the average data rate received across all nodes is only 45% of the source rate when nodes upload no data and 47% when nodes delay some data.

We wish to emphasize that our focus in this chapter is on mesh-based approaches for live video streaming, rather than file-download systems like BitTorrent [1]. While some of the attacks we consider may also be relevant to file-download systems, the impact on application performance is far more serious for streaming applications given that they are associated with stringent real-time deadlines. Consequently, the solutions must also be tailored to the unique demands of streaming applications.

Our contributions are:

- We provide a taxonomy of the implicit commitments made by nodes when peering with others. We show that when these commitments are not enforced explicitly, they can be exploited by malicious nodes to conduct attacks that degrade the data delivery service. To our knowledge, this is the first effort at taxonomizing attacks on mesh-based streaming protocols.

- We present a novel reputation scheme that combines feedback from the data plane (based on data received from the nodes) and the control plane (based on who a node has as neighbors) to increase the robustness of the mesh-based streaming overlay to the identified attacks. Through detailed security analysis, we show that our scheme is resistant to attacks commonly associated with reputation schemes such as self-promotion and slandering [107]. In particular, we show that our scheme ensures that a malicious node must contribute a minimum amount of data in a timely fashion to acquire a certain reputation. In addition, we show that a benign node that contributes data is assured a certain minimum reputation and cannot be slandered.

- We further augment the system, with a more comprehensive approach that also addresses potential vulnerabilities in the bootstrap mechanism, and with the source of the broadcast. We present a set of simple mechanisms to achieve this goal. Specifically, we present a scheme that prevents malicious nodes from influencing the membership bootstrap service and a source protection scheme that disallows malicious nodes to be overly connected to it.

- We evaluate our design using experiments on the PlanetLab testbed. Our results show that our schemes are extremely effective in ensuring good performance under attacks. With the local-reputation scheme, with 10% of the nodes being malicious, the average data-rate received across nodes from the source increases from 45% to 65%. Augmenting the solution with source and bootstrap protection mechanisms results in nodes receiving 95% of the source-rate on average. Our schemes also work well when attackers use advanced techniques such as data delaying. In fact, even with 30% of the nodes being malicious, more than 85% of the peers receive over 90% of the data. Overall, our results show the feasibility of augmenting mesh-based P2P streaming schemes to be resistant to attacks that target data delivery.

The rest of the chapter is organized as follows. In Section 5.2 we describe the mesh model we consider for this work. We describe attacks against data delivery in meshes in Section 5.3 and present our design to mitigate such attacks in Section 5.4. In Section 5.5 we provide an analysis for the security of our design. We explain the methodology and results of our experiments in Section 5.6 and 5.7, respectively. Finally, we summarize our findings in Section 5.8.

5.2 Mesh-Based Peer-to-Peer Streaming

We consider a unidirectional mesh-based P2P overlay consisting of a bootstrap node, a source node and peer nodes. As seen in Figure 2.2, the mesh allows peers to download a stream generated by the source, while the bootstrap maintains a list of alive peers used to assist peers to join the network. We consider a unidirectional mesh since it is more general than a bidirectional mesh. Also, unidirectional meshes have been shown to perform better than bidirectional meshes [13].

Every peer node maintains two sets of nodes, in-neighbors and out-neighbors. The in-neighbors represent the nodes that the peer node is receiving data from. The size of the in-neighbors is a system parameter. The out-neighbors represent the nodes that the peer node is sending data to. Each node decides independently the number of out-neighbors to support which will be proportional to its bandwidth. The source has no in-neighbors, only an out-neighbors set, whose size is usually larger than the size of an out-neighbor set of a peer node.

At join time, a peer node j first contacts the bootstrap node to receive a set of candidate nodes to serve as its neighbors in the overlay. Node j then contacts each candidate node and requests to become one of its out-neighbors. If a candidate node c accepts the request, then in turn, j will add c to its in-neighbor set. Each node pro-actively looks for several out-neighbors to connect to as well.

After it joins the overlay, a node discovers other peers by occasionally contacting its neighbors to learn about their own neighbors. This gossip protocol allows a node to update its in-neighbor set when neighbors leave or crash. A node also registers with the bootstrap node occasionally to allow the bootstrap node to have an up-to-date list of alive nodes. We will refer to these protocols as the *control plane* of the overlay.

The source node splits the stream into data chunks of a fixed size, each uniquely identified by a sequence number. To receive a chunk a node will send a request to an in-neighbor with that chunk's sequence number. If the requested node does not respond before a deadline then the requesting peer will consider that request lost. Each peer node maintains a buffer that it is trying to fill with data chunks. The buffer corresponds to a playback deadline, such that if a block of the stream is not received before that deadline, the data is considered lost and thus the quality of the playback stream is diminished. We will refer to this protocol as the *data plane* of the overlay.

This model is general enough to capture the characteristics of several previously proposed and deployed mesh-based systems [7, 12, 86, 97, 98].

5.3 Attacks Against Data Delivery

We state the assumptions we make about the attacker and provide a taxonomy of attacks against mesh-based P2P streaming systems.

5.3.1 Attack Model

We assume that a fraction f of peers are compromised and can behave arbitrarily. The percentage f is the largest fraction of nodes that the system is willing to tolerate as malicious. Their main goal is preventing the overlay from delivering data to each peer in a timely fashion. An attacker can disrupt the data delivery *directly* by attacking the data plane, or *indirectly* by attacking first the control plane to gain control over the data delivery path and then disrupting the data delivery.

We assume a defense against Sybil attacks [108] is in place, such as binding IP addresses to certificates or one that leverages social networks [109]. We also assume that data integrity is ensured and data is protected from pollution [82, 101]. We assume that the source and the bootstrap node are trusted and always behave correctly.

5.3.2 Attacks on the Data Plane

When two nodes A and B accept each other as out-neighbor, and in-neighbor, respectively, they assume several implicit commitments from each other:

• **Data delivery commitment:** A commits to B that it is going to deliver a certain amount of data to B.

• **Data download commitment:** B commits to A that it is going to download a certain amount of data from A.

• **Data upload commitment:** B is going to upload to the overlay what it downloaded from A.

• **Source upload commitment:** If B is connected to the source, then it will upload the data downloaded from the source to others in the overlay. This is similar to the data upload commitment, however we list it separately given that the source is a special entity where all the data originates.

• **Data delay commitment:** A will upload the data requested by B as soon as possible and not arbitrarily delay it.

• **Data integrity commitment:** A commits to B that it is not going to upload to B meaningless data.

However, in many mesh systems, not all of these commitments are explicitly enforced by the system. As a result, malicious nodes can exploit them to attack the data plane. We identify the following attacks (summarized in Table 5.1).

• **Data dropping attacks:** If the data delivery commitment is not met, a malicious node can accept benign nodes as its out-neighbors, but not deliver data to them. The attacks are effective because each data chunk has a strict deadline. A node only has time to make a few downloading attempts for a chunk, and will miss it once the deadline is passed.

• **Data delaying attacks:** If the data delay commitment is not met, a malicious node can send data to its out-neighbors yet delay the sending of it. Delaying data makes the attacker seem less malicious since it is actually delivering data before the

Table 5.1
Attacks against data and control planes

Data plane	Data dropping
	Data delaying
	Neighbor exhaustion
	Source
	Free-riding
	Pollution
Control plane	Bootstrap list pollution
	Neighbor selection

playback deadline. However, the data is less useful to the recipient since there will be fewer opportunities to upload the data to others.

• **Neighbor exhaustion attacks:** If the data download commitment is not met, a malicious node can become out-neighbors of benign nodes, but not download data from them. As many meshes limit the number of out-neighbors to ensure that nodes can honor the bandwidth requirements, by being included in the out-neighbors a malicious node exhausts the slots in that set thus denying access to other benign nodes.

• **Source attack:** If the source upload commitment is not met, malicious nodes do not forward data given to it by the source. Thus, if a particular chunk is only received by malicious nodes it will not be available to any benign node. To amplify this attack, malicious nodes can also become out-neighbors of benign nodes connected to the source and similarly not forward data given to them.

• **Free-riding attacks:** If the data upload commitment is not met, malicious nodes could also download data but not upload them to other peers, and basically obtain free service without contributing to the system.

• **Data pollution attacks:** If the data integrity commitment is not met, malicious nodes can upload meaningless data, thus polluting the information in the overlay.

5.3.3 Attacks on the Control Plane

The above data plane attacks are more effective when they impact many nodes in the overlay. A malicious node can increase the impact of its attack by first attacking the control plane. The control plane provides nodes with two mechanisms to discover peers. The first consists of the list of alive peers provided by the bootstrap node when a node joins the overlay. The second consists of exchanging membership information between the node and known peers. The bootstrap list is up-to-date if peers periodically register with the bootstrap node to inform it that they are alive. Assuming the bootstrap node is trusted, the control plane achieves its goals if the following commitments are met:

• **Registration with the bootstrap node commitment:** A peer commits that it will register occasionally with the bootstrap node, at a rate specified by the protocol.

• **Referral list commitment:** A node commits to provide a neighbors list that does not purposely contain malicious nodes and is not biased towards some nodes.

We identify the following attacks that have an impact on neighbor selection:

• **Bootstrap list pollution attacks:** If the registration with the bootstrap node commitment is not met, malicious nodes can register fast and often with the bootstrap node filling the bootstrap node's list of alive peers. Thus, although the bootstrap node is trusted, the list that it will provide to the joining peers will be polluted with malicious nodes. Note that malicious nodes can also register infrequently or not at all, but in this case they will not impact the list of the bootstrap node.

• **Neighbor selection attacks:** If the referral list commitment is not met an attacker can collude with other malicious nodes and when contacted about its own neighbors, refers only other malicious nodes. This attack is epidemic in nature since soon benign nodes will also be referring the malicious nodes they know to other benign nodes.

5.3.4 Our Focus

We focus on the attacks that we believe can be the most effective strategy for an attacker to disrupt the data delivery, and allow him to inflict maximum damage on the system with minimal resources. The most effective strategy for a malicious node is to (i) become neighbors of as many nodes as possible, (ii) deliver as little data as possible and (iii) data that is delivered should be as useless as possible. Hence we focus on control plane attacks (i.e. bootstrap list pollution and neighbor selection) that seek to increase the connectivity of malicious nodes and also on several data plane attacks (i.e. dropping, delaying, neighbor exhaustion, and source) as they can create considerable damage in the network.

We note that many of these attacks are specific to streaming, as file-distribution systems do not have real-time deadlines of data, nor need to download at a particular streaming rate, and often have centralized membership protocols (e.g. BitTorrent).

We do not consider attacks such as free-riding or data pollution as they relate to selfish behavior and data integrity but not attacks on data delivery. Furthermore, several solutions to free-riding have been proposed in previous work [1,103,104]. Also, to prevent data pollution, Dhungel et al. [101] have shown that a suitable means to accomplishing this is the source digitally signing hashes of the chunks. We note that solutions to these attacks can be used to complement our work.

5.4 A Design For Securing Data Delivery

In this section we describe our design for securing the data delivery for a P2P mesh-based streaming overlay. We first outline the design goals, then describe the details of our design.

5.4.1 Design Goals and Overview

Our focus is on ensuring that the P2P system achieves its intended goal which is continuous data delivery, even when under attack. However, achieving the same level of service in the presence of insider attacks as in the benign case is not always possible. As a result, our specific goals are:

(G1) Limit the impact of the attack: We seek to raise the bar for the attacker and bound the amount of damage per attacker. The damage created is directly proportional with the number of attackers and the amount of data dropped or delayed by the attacker nodes. Our goal is to limit the degree of connectivity in the mesh that malicious nodes can obtain. We integrate mechanisms that use control plane feedback to mitigate bootstrap list pollution, source, and neighbor selection attacks and mechanisms that use data plane feedback to detect data dropping, delaying and neighbor exhaustion attacks.

(G2) Limit the overhead of the defense mechanisms: Because malicious behavior is not a priori known, some of the components of our design are proactive, thus they must be enabled regardless of the presence of attacks. One specific concern is the overhead of the defense mechanisms. Our goal is that when no attack takes place, the system performance with the defense mechanisms enabled is the same as if those defense mechanisms were not used.

To achieve the goals identified above we design several proactive and reactive protocols. Our schemes use local observations to help nodes identify malicious peers and build a robust neighbor set. We also design schemes tailored for the source and bootstrap nodes given their critical roles.

Peer protection: To limit the impact of attacks and the overhead of the defense solution, we use decentralized mechanisms deployed at each individual peer that allow it to make local decisions about accepting, rejecting, or excluding other peers from its set of neighbors. Each node individually derives reputation scores for the other peers it is aware of in the overlay. The use of reputation is a natural choice in a distributed system with malicious participants. Since many existing reputation systems require additional overlays or have high computational or bandwidth overhead [110], we design schemes that are tailored to streaming overlays. The novelty of our scheme lies in combining feedback from the data plane and control plane to build reputations for each peer.

Source protection: As the source is a producer but not a consumer of data, the protection mechanisms used for peers are not applicable to the protection of the source. We use mechanisms that limit the impact of source attacks by allowing nodes to notify the source if certain data was not received.

Bootstrap protection: The bootstrap node plays a critical role in the control plane. Attacks against the control plane can be amplified if the bootstrap is not a reliable and unbiased source of information on who is currently in the overlay. Our scheme discourages nodes from registering at a fast rate and thus limits the percentage of malicious nodes in the bootstrap list.

Below we describe in details each of these protection mechanisms. First, we describe in Section 5.4.2 the details on a local reputation mechanisms that protects against data dropping and data delaying attacks. Then, we describe the source and bootstrap node protection mechanisms, in Sections 5.4.3 and 5.4.4, respectively.

5.4.2 Protecting Peers through Local Reputation

We propose a mechanism that allows peers to select as neighbors the nodes that provide the best performance while being resilient to data dropping and neighbor selection attacks. We also show how to extend this mechanism to protect against data delaying attacks. A node uses locally observed data and control plane information to compute scores for each of its neighbors. The lower the score, the higher the chance that a node is malicious. Nodes that have a score lower than a threshold T_d are evicted from the in-neighbors set. The local reputations are also sent across one hop to neighbors, so that they can avoid accepting malicious nodes as in-neighbors. The score consists of two components:

• *Data score:* This score is a *positive reputation* (it rewards good behavior) and it is calculated based on how much data a node has received from a particular neighbor. The goal of the data score is to capture regular performance degradation and data dropping attacks. Nodes who do not deliver sufficient data will have a lower data score. Nodes with a data score below a threshold T_s are considered to be *suspicious.* This approach forces malicious neighbors to deliver a certain amount of data. Note that for a node to be evicted from the neighbors set, his total score has to be smaller than T_d ($T_d < T_s$).

• *Graph connectivity score:* This score is a *negative reputation* (it penalizes bad behavior) and it is calculated based on how connected a node is to other nodes. The goal of the graph connectivity score is to target neighbor selection attacks. This score is relevant only for suspicious nodes because if the nodes deliver enough data (i.e. corresponding to a data score above T_s) they do not disturb the overlay. A high graph connectivity score indicates that a node is potentially conducting a neighbor selection attack. This score is used because if the data score is neither high nor low, it may not be obvious if a node is malicious.

Below we provide details about the data and graph connectivity score computation, about the way they are combined into a reputation score, and about how the reputation score is used to make decisions on what nodes to allow as neighbors. Algorithm 2 also describes this computation, specifically how a node i calculates the *Local Reputation* for node j.

Data score computation. Every node i calculates a data score for every in-neighbor j as follows:

$$L_{ij}(t) = min\left(1, \frac{G_{ij}(t)}{E(t)}\right) \tag{5.1}$$

where $G_{ij}(t)$ is the number of chunks received by node i from j before deadline D_r in a time period. D_r is the amount of time the requesting peer will wait before considering that the request was dropped. If a request for a chunk is honored after the D_r deadline has passed, it is not included in $G_{ij}(t)$. $E(t)$ is the expected number of chunks to be received by a node in a time period. Typically, the expected value is the same for all nodes and if it is received from all in-neighbors the full streaming rate will be received. We take the minimum of $\frac{G_{ij}(t)}{E(t)}$ and 1 so that if a node performs better than expected the end result will still be between 0 and 1. The more data j delivers to i, the bigger the $L_{ij}(t)$. If $L_{ij}(t)$ is less than a threshold T_s, then i marks j as *suspicious*.

A score for a node's out-neighbors is calculated by replacing $G_{ij}(t)$ with the number of requests fulfilled for that node in a time period. Such a score allows nodes to mitigate neighbor exhaustion attacks.

Graph connectivity score computation. Every node also calculates a graph connectivity score for each of its neighbors that were marked suspicious. This score relies on the observation that a malicious node conducting a neighbor selection attack will be an in-neighbor for many honest nodes. In particular, if node i has as in-neighbors node k and j, and node j is malicious, then it is likely that j is an in-neighbor for node k as well. Furthermore, the more in-neighbors of i that j is connected to, the more likely it is that j is conducting an attack. We propose the following graph connectivity equation for each node i to calculate the likelihood of each of its in-neighbor j being malicious:

$$C_{ij}(t) = \frac{K_{ij}(t)}{N_i(t)} \tag{5.2}$$

where $N_i(t)$ is the total number of non-suspicious neighbors of i (i.e. a non-suspicious neighbor is one whose data score L is greater than T_s), and $K_{ij}(t)$ is the number of these non-suspicious neighbors for whom j is also an in-neighbor. Intuitively, the equation calculates a score equal to the percentage of non-suspicious neighbors that a neighbor j is currently an in-neighbor for. The score will be high if a neighbor is in many neighbor sets, indicating that it is malicious. We consider only non-suspicious nodes so that in the case a malicious node wants to falsely advertise other nodes in its in-neighbor set, it has to first perform some work for the system.

Reputation score computation. Every node combines the data and graph connectivity score as follows:

$$R'_{ij}(t) = \begin{cases} L_{ij}(t) - \alpha * C_{ij}(t) & \textit{if } j \textit{ is suspicious} \\ L_{ij}(t) & \textit{otherwise} \end{cases} \tag{5.3}$$

If a node had a low data score and was marked as suspicious, then we take into account the graph connectivity score as it is a negative score and will further reveal if the node is misbehaving. Specifically, we subtract from the data score the graph connectivity score and we weight the latter with a parameter α. However, if the node was not marked as suspicious, we do not take into account the graph connectivity score. This choice was made based on the observation that if the nodes deliver enough data, it does not matter how connected they are as they do not disturb the honest nodes.

Incorporating history. Every node takes into account the history of its neighbors by calculating for each neighbor the following equation:

$$R_{ij}(t) = \lambda * R'_{ij}(t) + (1 - \lambda) * R_{ij}(t-1) \tag{5.4}$$

where λ is a value less than 1. We take into account history to accommodate transient network conditions, such as congestion. This gives nodes the opportunity to recover and not be disconnected due to non-persistent problems. All nodes start with a reputation equal to T_s.

Reputation based neighbor selection. A node uses reputation scores to decide when to drop or add neighbors. To decide if he keeps a node j as a neighbor, node i compares the reputation score R_{ij} for node j with a threshold T_d. If j's score becomes less than T_d, then i will drop j from its neighbor set and will not allow j to be in either its in-neighbor or out-neighbor sets from then on. We identify j by its IP address to avoid trivial Sybil attacks.

A node also uses the reputation score to determine if a node is non-malicious when deciding to add a neighbor. Consider the case when a node s refers a neighbor k to node i, s will also send the reputation score of k. To decide if he adds k as a neighbor, i computes $R_{is} * R_{sk}$. Node i will then add node k as a neighbor if the resulting number is greater than the suspicion threshold, T_s.

Protecting against data delaying. As long as there is no enforcement of data delivery, an attacker's best strategy is to drop all data. However, once the above protection mechanisms are introduced, an attacker's best strategy is to send as little data as possible, but delay everything it sends so that the node receives it just before the D_r deadline. The delaying of data is advantageous to attackers in multiple ways. Attackers can still get credit for sending data, yet it is less likely that the benign node will have as many opportunities to pass that data on to others. Delaying data also creates a temporary scarcity of data chunks, and as few benign nodes will have that data, malicious nodes will be able to fill that request for many nodes.

As data delaying unnecessarily increases the amount of delay in receiving chunks, nodes can measure the delay and then penalize the offenders. To do this we introduce the inverse relative stretch (IRS_u) metric which node i will calculate for each chunk u received from node j. We define IRS_u as the ratio between the delay from the source to node i and the delay from when the source generates a chunk u to when node i actually receives it. An IRS_u of 1 would indicate node i received the chunk with no extra delay whatsoever while less than 1 indicates that there was some extra delay. To incorporate this value into the data score, we recalculate G_{ij} (as referenced in Equation 5.1) as follows:

$$G_{ij}(t) = \sum_{u} min(l * IRS_u, 1) \tag{5.5}$$

During one time period, node i evaluates the IRS_u of every chunk u received from node j and calculates a summation based on these values. Specifically, the summation of G_{ij} is calculated by adding the minimum of 1 and l times IRS_u for every chunk received. We multiply the IRS_u by some parameter l as some stretch is normal for any application-layer multicast and l lets us determine how much stretch we are willing to tolerate. We then take the minimum of that value and 1 to normalize it and ensure that we are adding at most 1 for every chunk received. We would expect then that a benign node that does not add extra delay to a chunk would receive a score equal to the number of chunks it sent. However, any malicious node adding extra delay would receive a lower score.

Overhead. The *Local Reputation* scheme adds minimal overhead to the system. This is due to it only involving simple calculations, the most expensive being multiplications, and it scales linearly with the number of neighbors a node has, which typically runs between 15 and 60, depending on how much bandwidth a node has. Furthermore, the network overhead is negligable as we use messages that are already being sent to transport the extra data about reputation scores.

5.4.3 Source Protection with Health Monitoring

The source is a critical component of the overlay. As will be shown in Section 5.7 attacks against the source can significantly degrade the performance of the system. While *Local Reputation* is effective for peers, it can be intuitively seen that such a mechanism is ill-suited for the source. This is for two reasons. First, the source does not request data from its neighbors, it only gives data, so it cannot judge a node based on data received. Second, malicious nodes will prefer to receive data from the source rather than from peers so this also will not lead to the source suspecting them. As a result the source can not differentiate between a benign and malicious node.

We first observe that in some P2P live streaming systems today, there is extensive gathering of statistics from peers [82]. This allows for further refinement of protocols and code so that the quality of the experience can continue to improve. One very important metric to collect is the amount of data that peers miss from the stream. This gives a way to measure the overall health of the system. We then use this monitoring information to protect the source.

Specifically we propose that the source keeps track of who it sends which data chunks to. Then if peers miss some data chunks, they can report the specific ones missed to the source. One would expect that if many nodes miss a chunk, it is due

Algorithm 2: *Local Reputation* computed by node i for node j.

```
//This algorithm is run every $t$ seconds
//$R_{ij}$ is initialized to be $T_s$ when node $j$ becomes a neighbor
$G_{ij} = 0$;
$E$ = source_streaming_rate / num_neighbors;
//compute data score
foreach chunk $u$ sent by $j$ do
    $G_{ij} += 1$;
    // if protecting against data-delaying do instead:
    // $G_{ij} += min(l * IRS_u, 1)$;
end
$L_{ij} = min(1, G_{ij}/E)$;
if $L_{ij} < T_s$ then
    //node is suspicious, consider graph connectivity score
    $N_i = 0$;
    $K_{ij} = 0$;
    foreach in-neighbor $k$ of $i$ do
        if $L_{ik} > T_s$ then
            // in-neighbor $k$ is not suspicious
            $N_i += 1$;
            if $j$ is in-neighbor of $k$ then
                $K_{ij} += 1$;
            end
        end
    end
    $C_{ij} = N_i / K_{ij}$;
    //reputation considers both data and
    // graph connectivity scores
    $R'_{ij} = L_{ij} - \alpha * C_{ij}$;
else
    //reputation only considers data score
    $R'_{ij} = L_{ij}$;
end
//take into account history of reputation
$R_{ij} = \lambda * R'_{ij} + (1 - \lambda) * R_{ij}$;
if $R_{ij} < T_d$ then
    //node $j$ is below drop threshold
    disconnect $j$
end
```

to malicious nodes not forwarding data received from the source. Therefore, once the source has received complaints from a percentage of nodes greater than f it can then disconnect the nodes it sent those data chunks to. The percentage f must be the largest fraction of nodes less than 50% that the system can tolerate as malicious. Also, as this scheme might create an implosion of messages at the source, nodes can collect them and send them in batches.

Overhead. In practice, the network overhead is small as we batch complaints into a single message and only send them periodically. Furthermore, the source will only receive messages when nodes are not receiving the data chunks, which should be abnormal behavior. To further reduce the overhead, nodes could piggy-back the complaints on messages that are already being sent to the bootstrap.

5.4.4 Rate-limiting Bootstrap

Our solution for protecting the bootstrap relies on the observation that nodes that register with the bootstrap node many times in a short period are most likely malicious. Thus to detect and discourage this behavior, if nodes register faster than once every w seconds, they will not be put into the bootstrap list and then will not be propagated by the bootstrap node. To detect misbehavior the bootstrap keeps track of all registrations that have occurred in the past w seconds. From the registration information it will make a list of k nodes that have only registered once.

The w parameter decides how often nodes can register, so the larger it is the more resilient the bootstrap will be against attacks. However, if it is too large it will prevent good nodes from legitimately re-registering. The k parameter allows the bootstrap to decide how it will pick nodes from the recent time window and in what quantity. There are different strategies to fill the bootstrap list, for our design though, we simply choose the k most recently registered nodes to ensure the freshness of the list.

To only do rate-limiting and nothing else might bring about scenarios where there are still very few honest nodes in the bootstrap list. This could be due to very few nodes joining the overlay for a period of time. To ensure that the bootstrap list still can not be filled with malicious nodes, we have each node randomly register once every w to $2w$ seconds.

Overhead. The *Rate-limiting Bootstrap* scheme introduces no new overhead into the system, as periodically registering with the bootstrap is normal behavior. Additionally, the rate of at which registrations occur, and thus the amount of overhead, can be adjusted by setting w to the desired rate.

5.5 Security Analysis

In this section, we analyze how robust the *Local Reputation* scheme is in defending against common classes of attacks. Recall that the final reputation score is derived by combining the data score, which is a positive score, and the graph connectivity score, which is a negative score. The node uses the final reputation score to decide who should remain as neighbors and who to admit as neighbors. Possible attacks that can be conducted on these reputation calculations and uses include [107]:

Self-promoting: Malicious nodes falsely inflate their own reputation. This attack is only effective in positive feedback based systems.

Slandering: Malicious nodes attack the reputation of other nodes by reporting untrue information about them. This attack is only effective in negative feedback based systems.

Orchestrated: Colluding nodes combine several strategies to game the system.

Whitewashing: Malicious nodes take advantage of a system vulnerability to restore a damaged reputation. One possible way to do this is by assuming new identities.

5.5.1 Attacks on Data Score Calculation

The reputation system is designed so that a node cannot get a high data score and thus a high reputation without doing useful work. Therefore, the data score cannot be influenced by slandering or self-promoting attacks, as the only way to change it is for a node to deliver more data. We present the following lemma which quantifies the amount of useful work done by a node given a particular data score, which can be derived from Equation 5.1.

Lemma 1: *For a node j to obtain a data score of L_{ij} at a neighboring node i, j must deliver data to i at a minimum rate of $E * L_{ij}$, where E is the expected amount of data a node should deliver to a neighbor in a time window (Section 5.4.2).*

This lemma guarantees that benign nodes will receive good performance even when surrounded by a significant number of malicious neighbors, for example, when under an orchestrated attack. This is because each malicious neighbor is forced to deliver a minimum amount of data in order not to be dropped. More specifically, if we assume a node with a fraction f of its neighbors is malicious, and assume benign neighbors always deliver the expected amount of data, then the node will receive at least $(1 - f) + T_d f = 1 - f(1 - T_d)$ of the streaming rate (T_d is the drop threshold). For example, with $T_d = 0.5$ and $f = 0.3$, the node will receive at least 85% of the stream rate.

Furthermore, Lemma 1 imposes a high bandwidth cost on malicious nodes who seek to be a neighbor of a large number of nodes. To highlight this, consider a streaming system with 150K nodes [82], and that a malicious node desires to maintain a reputation score of T_d at every node. According to Lemma 1, with a streaming rate of 1Mbps, a neighbor-set size of 15, and assuming a T_d value of 0.5, the node must deliver data at a minimum total rate of 5Gbps.

We note that though Equation 5.5 modifies how the data score is calculated to protect against data delaying, this simply raises the bar for attackers, forcing them to send even more data if they wish to delay the data they are sending. Hence, even when data delaying protection is in place malicious attackers still must send at minimum $E * L_{ij}$. We present the following lemma which quantifies how much more data a node must send if it delays it.

Lemma 2: *For a node j to obtain a data score of L_{ij} at a neighboring node i when delaying data, j must deliver data to i at a minimum rate of $\frac{E*L_{ij}}{l*IRS}$, where IRS is the inverse relative stretch and l is a system parameter.*

Lemma 2 shows that attackers must increasingly send more data the longer they delay it. For example, if node i and the source have a delay of 100 ms and node j delays data so that it arrives after 600 ms, the IRS will be 1/6. Assuming l is set to 3, then node j must send data at a rate of $2 * E * L_{ij}$ to get a score of L_{ij}, in this case doubling the amount of data it would normally have to send.

5.5.2 Attacks on Graph Connectivity Score Calculation

When a node calculates its neighbors' graph connectivity score, it takes into account neighbor set information provided by all of its non-suspicious neighbors. This scheme is subject to slandering attacks where a malicious neighbor can provide fake neighbor set information. Slandering can be seen from two different perspectives, the ability of a node to slander others and the resistance a node has from slandering attempts. We first present the following lemma that shows the limitations a node has in its ability to slander others, which can be derived from Equation 5.2.

Lemma 3: *A node j can only influence the graph connectivity scores of the neighbors of node i if j has a data score of T_s with i.*

Lemma 3 shows that malicious nodes must themselves do a substantial amount of work to remain non-suspicious, which means having a data score above T_s. According to Lemma 1, this means they must deliver data to i at a minimum rate of $E * T_s$. Given that $T_s > T_d$, this imposes an even greater bandwidth constraint on attackers that want to slander others over attackers that want to simply not be dropped.

We next present a lemma that demonstrates that a node can resist slandering attacks from others. The key insight behind the lemma is that if a node transmits data at a high enough rate, its final reputation as computed by the neighbor depends on the data score alone, and is not impacted by the graph connectivity score.

Lemma 4: *Any node that delivers data to a neighbor at a rate greater than $E * T_s$ is assured of a reputation greater than T_s with the neighbor.*

We expect that most benign nodes will be cooperative and deliver data at rates close to the expected rate, which is well above $E * T_s$. Therefore, benign nodes will not be subject to slandering attacks as their graph connectivity score will not even be considered.

5.5.3 Other Attacks

We discuss other attacks on the schemes, and why our approach is resilient to them:

Whitewashing attacks: In these attacks, malicious nodes who received a bad reputation may choose to rejoin the network with a different identity. We believe this attack is not a concern because of the following reasons. First, the reputation is initialized to T_s, and all new nodes will be marked suspicious initially. Therefore, a new node cannot refer other nodes or report connectivity information about other nodes until it has done work and improved its reputation. Further, the newly added node will be quickly dropped unless it transmits data at a sufficient rate. Second, in our model, nodes are identified by their IP address. To cause damage, a malicious node cannot acquire a new identity by simply spoofing an IP address, but must be able to receive packets targeted to the IP address. By our attacker model in Section 5.3.1, we assume only a fraction f of the total number of IP addresses are controlled by malicious nodes.

Attacks on reputation-based neighbor selection: A node adds new neighbors by taking referrals from existing non-suspicious neighbors. This process is subject to attacks where a malicious neighbor (m) could refer other malicious nodes to a benign node (i). However, to conduct this attack, the malicious neighbor m must be considered non-suspicious, and hence must deliver data at a minimum rate of $T_s * E$, where T_s is the suspicion threshold. Further, each newly inserted malicious node referred by m must also do a substantial amount of work to obtain a minimum data score of T_d (the drop threshold), or it will be dropped quickly by node i.

5.6 Experimental Methodology

In this section, we describe how we evaluate and compare our protection schemes with several alternative schemes. We implemented the unidirectional mesh described in Section 5.2 in a mesh streaming codebase [84]. We also implemented all of our own protection schemes plus some alternatives which we will describe next, which are summarized in Table 5.2.

5.6.1 Schemes Considered

Table 5.2
Mechanisms for each component of system

Peers	Source	Bootstrap
Least Performing Peer (LP)	Drop Periodically (DP)	Periodic Register (PR)
Local Reputation (LR)	Health Monitoring (HM)	Rate-limiting (RB)
LR Data Delaying (LR-DD)		

No Protection (NP): This is our baseline scheme which has no protection for any of the system components.

Local Reputation (LR): This peer level scheme, as described in Section 5.4.2 builds up a reputation from information gathered from the control and data planes. With this reputation scheme in place, nodes are able to decide if a node is malicious and thus can better select who they should accept as neighbors.

LR Data Delaying (LR-DD): This is the extended Local Reputation scheme that adds protection against data delaying attacks.

Health Monitoring (HM): This source protection scheme, described in detail in Section 5.4.3, uses information gathered from peers to decide who should stay as neighbors of the source. If a percentage of nodes declare that a certain data chunk was missed by them, the source will drop the nodes that it originally sent those chunks to.

Rate-limiting Bootstrap (RB): This is a bootstrap protection scheme, described in detail in Section 5.4.4, keeps track of how often nodes register and penalizes the ones who register fast. The bootstrap will only refer nodes who register at a rate less frequently than the rate it specifies.

Least Performing Peer (LP): This is a peer level scheme, similar to the one used in CoolStreaming [86], that drops the in-neighbor that is currently contributing the least amount of data. We chose this alternative to LR because of its simplicity and to show that while simple schemes such as this prove to be effective in a setting where all nodes are benign, more robust methods are needed when malicious nodes are present.

Drop Periodically (DP): This is a source protection scheme that induces churn [111] on the source. We note that as time progresses and benign nodes churn in and out of the system, malicious nodes can continue to stay and eventually eclipse the source as its neighbors. To address this problem we allow a single node to stay as an out-neighbor for only a certain amount of time and then disconnect it. To further stagger the disconnection times of nodes, we only allow one node to be disconnected in a time period.

Periodic Register (PR): This is a bootstrap protection scheme that requires all peers to re-register every r time. We chose this scheme as an alternative to RB since it also requires re-registration of nodes, but does not do any rate-limiting. Thus, it demonstrates that more robust methods are needed when malicious nodes target the bootstrap service.

5.6.2 Attacks Considered

To show the effectivness of our schemes, we also implemented the most effective attacks available for an attacker to disrupt the data delivery.

• **Data dropping attack:** A malicious node will advertise the chunks of data that it has, but will not fulfill any requests made for those chunks unless otherwise stated.

• **Data delaying attack:** A malicious node will advertise and fulfill a small amount of chunk requests to avoid being dropped from another node's neighbor set. However, the malicious node will delay the sending of the chunk until very close to the chunk's deadline.

• **Source attack:** A malicious node will not forward data given to it by the source. As the source has limited bandwidth, if only malicious nodes receive a particular chunk, then that chunk will be effectively lost and no benign nodes will receive it.

• **Bootstrap list pollution attack:** A malicious node will register often with the bootstrap, to ensure that it is always in its list of peers and to increase its chances of being referred to benign nodes.

• **Neighbor selection attack:** When a benign node contacts a malicious node to discover more peers to connect to, the malicious node will bias its referrals to include only other malicious nodes. This results in benign nodes being neighbors with many malicious nodes.

5.6.3 Experiment Configuration

The experiments were run on the PlanetLab overlay testbed. The source was located on a host at our lab. We set D_r (see Table 5.3 for a list of parameters and their definitions) to be 1 second. We determined this value experimentally as we observed that in a non-malicious scenario 96% of nodes receive 99% of chunks within 1 second. Each node is configured to obtain up to 15 in-neighbors and the maximum number of out-neighbors is proportional to its bandwidth. The source will obtain 30 out-neighbors.

We used overlay deployments of 300 nodes. Each experiment lasted for 10 minutes. For each experiment we varied the percentage of malicious nodes from 0 to 30% and fixed the source's streaming rate at 1 Mbps. Each experiment was run for 10 times

Table 5.3
Notation

D_r	Deadline at which a peer considers a request for data dropped
T_s	The suspicion threshold
T_d	The drop threshold
α	When calculating $R'_{ij}(t)$ gives a weight to $C_{ij}(t)$
λ	When calculating $R_{ij}(t)$ gives a weight to the previous value of $R_{ij}(t-1)$ and the current value of $R'_{ij}(t)$
l	When using IRS to calculate $G_{ij}(t)$, defines how much stretch is to be tolerated

and the results were averaged. Standard deviations are plotted where appropriate. The malicious nodes joined at the beginning of the experiment and stayed for the entire duration. Benign nodes both join at the beginning of the experiment and also during the experiment. We modeled the join times by using a Poisson process and the participation time by a Pareto distribution. The mean of the Poisson process was 3 and the Pareto distribution is used with a shape parameter of 1.42, giving a mean participation time of 300 seconds and we also assume a minimum participation time of 90 seconds. The parameters have been used previously by Bharambe *et al.* [112] and were motivated by traces of real multicast systems [81] and Mbone sessions [113].

Choosing Parameters: For *Local Reputation* we by reason set its parameters to appropriate values and validated them experimentally. We set T_s to be 0.7 to tolerate transient network conditions. We note that T_s can be set by the user, to the minimum quality threshold that he is willing to tolerate. We set α to be 0.5 since we consider data plane feedback to be more useful than control plane feedback. We also conduct a sensitivity study of α in Section 5.7. For nodes to evict malicious nodes that are both suspicious and highly connected, the equation $T_d \geq T_s - \alpha$ must hold. Therefore we set T_d to be 0.2. We set λ to be 0.4 to give a greater weight to the history of the reputation but also be able to change quickly if nodes consistently behave badly. We set the time period for the recalculation of the scores to be every 3 seconds. For Local Reputation with Data Delaying protection (*LR-DD*) we experimentally set l to be 3. Therefore, the delay of the chunk must be 3 times greater than the delay to the source before a node is penalized.

5.6.4 Performance Metrics

We evaluate the effectiveness of the attacks and solutions with the following metrics.

Goodput Ratio: This represents the percentage of useful data a node received while in the overlay, averaged across all nodes. We use it to measure the effects of churn on the quality of the goodput. The higher the goodput ratio, the higher the quality of the stream received.

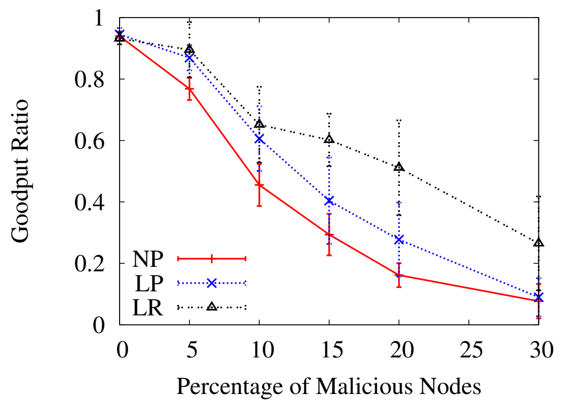
Corruption Factor: This represents the percentage of nodes in the neighbor set that are malicious. We use it to measure the level of control an adversary has on the neighbor set of a particular node. The higher the corruption factor, the more adversarial neighbors a node has.

5.7 Experimental Evaluation

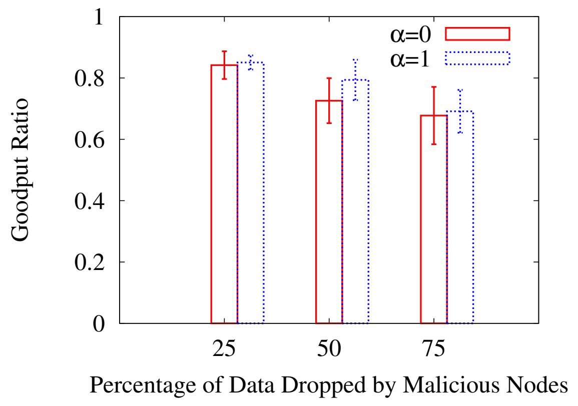
In this section we experimentally show that the schemes we proposed in Section 5.4 are able to effectively mitigate attackers.

5.7.1 Robust Neighbor Selection

To give motivation to our *Local Reputation (LR)*, we first compare it to *Least Performing Peer (LP)* and *No Protection (NP)*. Malicious nodes perform data dropping, source and neighbor selection attacks. As can be seen in Figure 5.1(a) both



(a) *No Protection*, *Least Performing Peer*, and *Local Reputation* schemes running at the peers with *No Protection* at the source.

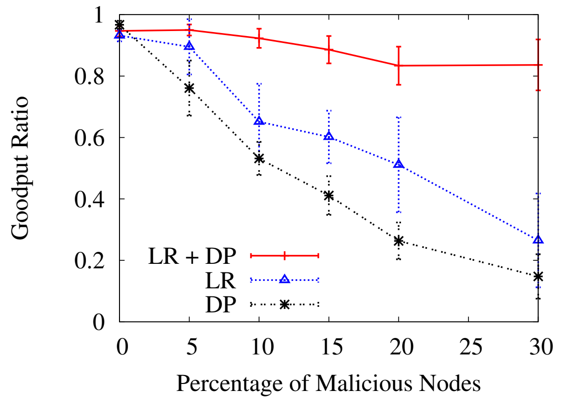


(b) Goodput ratio when 10% of participants are malicious and they drop varying amounts of the stream. *Local Reputation* is protecting the peers and α is varied.

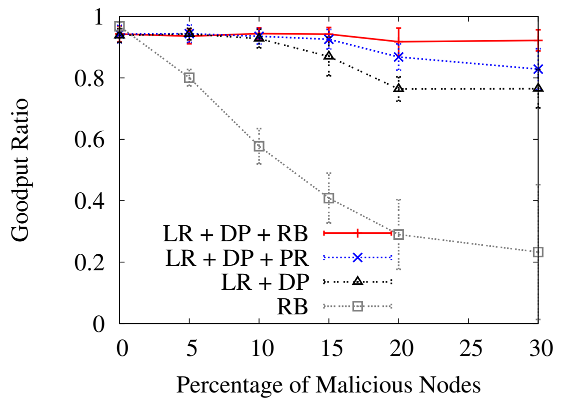
Figure 5.1. Importance of peer protection.

NP and *LP* perform worse than *LR*. This difference becomes more pronounced as the percentage of malicious nodes increases. *NP* is ineffective simply because nodes never change who their neighbors are, regardless of their poor performance.

LP is not as effective as *LR* since a node never drops all of the malicious nodes from its neighbor set. Further investigation shows that for a node running *LP* the



(a) Peers running the *Local Reputation* scheme with *Drop Periodically*, *Pretrusted Peers*, and *No Protection* at the source. For comparison we also plot the *Drop Periodically* only scheme.



(b) *Rate-limiting Bootstrap*, *Periodic Register*, and *No Protection* are combined with *Local Reputation* and *Drop Periodically*. *Rate-limiting Bootstrap* with *No Protection* is also shown.

Figure 5.2. Importance of source and bootstrap protection.

number of malicious nodes in its in-neighbor set decreases as some of the malicious nodes will be dropped. However, there are still malicious nodes present in the in-neighbor set because *LP* does not prevent the node from reconnecting multiple times to the same malicious nodes. When the node is running *LR*, it does not reconnect anymore to malicious nodes since malicious behavior is captured in the reputation score for those nodes.

Importance of considering graph connectivity: We examine the contribution of the graph connectivity score on *LR* and identify regimes in which its use is beneficial. We compare the case when the reputation score computation is based only on the data feedback (i.e. $\alpha = 0$) to the case when both data and graph connectivity are considered (i.e. $\alpha = 1$).

As we can see in Figure 5.1(b) when attackers drop 25% or 75% of the data they were expected to deliver, the performance does not change no matter the value of α. For the case of 25% dropping, recall that a node i will only calculate the graph connectivity score for a neighbor j if it marks j as suspicious (i.e. $L_{ij} < T_s$). When j drops 25% of the data it will not be marked as suspicious since we use a T_s value of 0.7, thus the graph connectivity score will not be considered. In the case of 75% dropping, enough data is dropped that the neighbor will be perceived as malicious by its data score alone. Hence graph connectivity is most useful in regimes where the amount of data dropped by a malicious node is large enough to be marked as suspicious, but not large enough to be interpreted as malicious by their data scores alone. This is the case for 50% dropping. In Figure 5.1(b), when attackers drop 50% of the data, *LR* combining the two scores performs better than *LR* using only data score. The information from the control plane about the existence of a neighbor selection attacks helps effectively identify malicious nodes.

We varied α even more to find values that give better performance but we found that a value of 1 is sufficient across all percentages of attackers.

5.7.2 Source Protection

While *LR* performs much better than other schemes, the goodput ratio achieved is still far from satisfactory. We believe this is because *LR* does not protect the streaming source, as we explained in Section 5.4. Further investigation into the source's performance confirms our hypothesis. While peers using *LR* expel all malicious nodes from their in-neighbor set, the source's out-neighbor set is almost full of malicious nodes. This illustrates the importance of having additional mechanisms to protect the source.

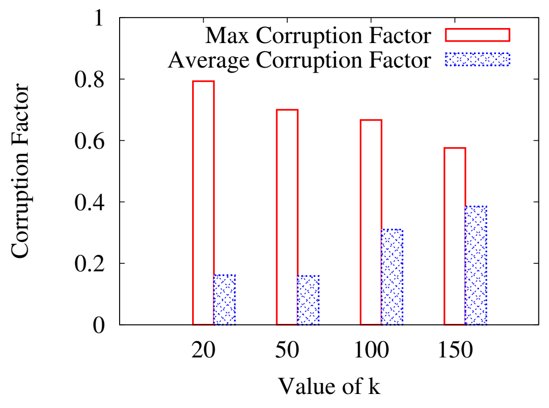
We next evaluate mechanisms that can be used to protect the source. When using *Drop Periodically (DP)* the source will drop a node after it has been a neighbor for 1 minute. In these experiments malicious nodes again perform data dropping, source and neighbor selection attacks. Figure 5.2(a) shows the results. The goodput ratio is significantly raised for *DP* combined with *LR* (i.e. the curve titled *LR+DP*). This is because *DP* effectively reduces the corruption factor at the source to a value that is very close to the percentage of malicious nodes in the overlay at all times, for all settings.

Figure 5.2(a) also shows that *DP* alone is not sufficient. This is not surprising, because *DP* protects only the source, not the peers. This again highlights that solutions must be employed at both the source and peers to achieve satisfactory performance.

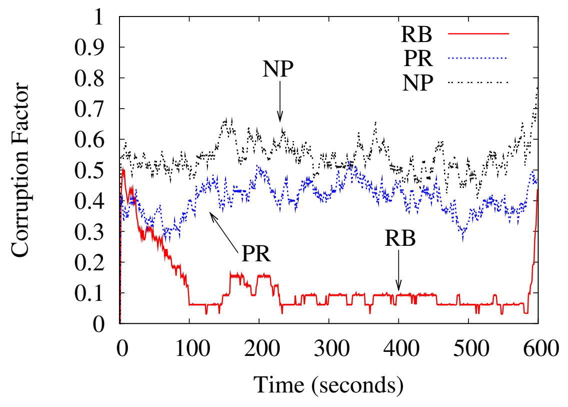
5.7.3 Rate-limiting Bootstrap

We now consider when malicious nodes also conduct a bootstrap list pollution attack, along with the data dropping, source, and neighbor selection attacks. We evaluate the effectiveness of *Rate-limiting Bootstrap (RB)* in mitigating such attacks and compare it with *Periodic Register (PR)*. Two parameters influence the performance of *RB*: the time period in which a node may register only once to be considered as non-malicious (w) and the size of the short list maintained by the bootstrap (k).

Selection of w: Taking into consideration the trade-offs described in Section 5.4.4, we set w conservatively at 300 seconds. This value is much smaller than the typical session length in P2P streaming systems, which is usually in the order of tens of minutes [81, 82, 114]. For *Periodic Register (PR)*, we use a w value of 120 seconds to



(a) Corruption factor at the source on a single experiment when 20% of nodes are malicious and they are aware that *Rate-limiting Boostrap* is running and do not register fast. We vary k to find the best value.



(b) Corruption factor at the source on a single experiment when 20% of nodes are malicious. We compare the effects on the source of the three different bootstrap node schemes.

Figure 5.3. Evaluating the corruption factor in different scenarios.

show that even when sacrificing overhead for a more up-to-date list and thus better security, rate-limiting schemes are still preferred.

Selection of k: We experimentally determine the value of k. We fixed the system solution to be *LR+DP+RB* and varied the value of k. The malicious nodes are aware of the solution and only register every w seconds in order not to get themselves excluded from the bootstrap's short list. The malicious nodes all register at the same time. Note that if they space out their registrations, the impact on the bootstrap list would be diluted. In Figure 5.3(a) there are four sets of bars each with a different k value, and two bars in each set corresponding to the maximum and average corruption factor at the source. The figure shows that as k increases, the maximum corruption factor decreases, but the average corruption factor increases. This is because the smaller the k, the easier it is for the attacker to flood the bootstrap's short list in a burst, thus achieving a high corruption factor at the source. However, each node will only remain on the list until k more nodes have joined. Thus, the larger the k, the longer a malicious node will stay on the list, resulting in a higher average corruption factor. From the figure we conclude that setting k at 50 is a good trade-off between having a large corruption factor all the time and having a large spike in the corruption factor every w seconds. In the rest of our experiments we set k to be 50.

Figure 5.2(b) shows the evaluation results. *RB* combined with solutions for source and peers (i.e. the curve titled *LR+DP+RB*) performs the best and mitigates the attack across all malicious percentages. *PR* combined with other solutions (i.e. *LR+DP+PR*) works equally well for small percentage of attackers (up to 15%). For higher percentages of attacker nodes, *PR* effectiveness decreases because the scheme simply puts both benign and malicious nodes on equal footing. Thus, while the bootstrap's list of nodes is very close to being up-to-date, it does not punish attackers. On the other hand the *RB* solution is more effective for exactly this reason, if nodes register too fast they are not made known to nodes who request a list of peers. We also note that *PR* incurs a large overhead at the bootstrap node as it requires all nodes to re-register with the bootstrap node often. Lastly, both schemes perform significantly better than *NP*, highlighting the importance of having additional mechanisms to protect the bootstrap node.

To gain more insight into these results, we also plot in Figure 5.3(b) the corruption factor at the source for each solution. Recall that *DP* at the source requires that the source only obtains neighbors from the bootstrap node, thus the degree of pollution at the bootstrap node directly affects the corruption factor at the source. The figure shows that the corruption factor is significantly lower with the *LR+DP+RB* than other solutions, further confirming its effectiveness.

5.7.4 Data Delaying

As evidenced by our analysis and experiments, attackers will be thwarted as long as they continue to drop data. To prolong the amount of time they can stay as neighbors and thus do more damage, attackers will necessarily have to actually give data to others. However, attackers are motivated to make sure the data that is given out is as useless as possible to benign nodes. Attackers can achieve this by delaying the sending of data to the last possible moment.

For the attack to succeed even though data is still being given away the attacker will need to make sure it has a good strategy for only giving out data that will become very common and not data that will remain rare. Attackers obviously do not know the future, but can assume that if no other benign nodes have the data then they should not pass it on to others, but if some other benign nodes do have the data, they can upload it to others.

To show how effective delaying is, we now run experiments where malicious nodes conduct data delaying instead of data dropping attacks. Malicious nodes also conduct source, neighbor selection and bootstrap list pollution attacks. We deploy the *LR-DD* and *HM* protection schemes and show how well they mitigate attacks. For *HM* we set the fraction f of nodes that the source must get complaints for before it disconnects a node to be 30%.

We present the results in Figure 5.4, which demonstrate that these attacks are effective in increasing damage done. For example, when peers and source are just protected with *LR* and *DP*, (i.e. the curve entitled *LR+DP+RB*) nodes increasingly have worse performance as the fraction of malicious nodes increases, culminating in

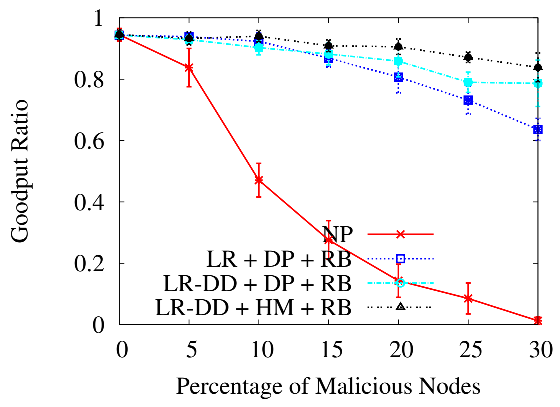


Figure 5.4. Peers running the local reputation scheme while attackers conduct data delaying attacks.

only a .68 goodput ratio when there are 30% malicious attackers. As *LR-DD* and *HM* protection schemes are added performance increases, effectively mitigating the attack. *LR-DD* proves to be effective as most benign nodes receive data fairly quickly after the source sends it out, thus malicious nodes delaying data are promptly removed from in-neighbor sets. *HM* also outperforms *DP* as it is able to actually identify malicious nodes who do not forward data to others and disconnect them, rather than simply keeping the fraction of malicious nodes low.

5.8 Summary

In this chapter, we present one of the first efforts aimed at systematically analyzing and addressing the vulnerabilities of mesh-based P2P streaming systems to malicious insider attacks. We consider both direct attacks on the data plane, as well as attacks on the control plane which could in turn lead to further disruption of data delivery. These include data dropping and neighbor selection attacks, as well as data delaying, which is a novel attack on P2P streaming. We present a design for securing data delivery, of which a key component is a reputation scheme that helps nodes identify malicious peers and build a robust neighbor set. Through detailed security analysis, we show that our scheme is resistant to a variety of attacks commonly associated with reputation schemes such as self-promotion, slandering, and white-washing [107].

We present an extensive evaluation of our design through experiments on Planet-Lab. Our results show that (i) without our solution, the data delivery can be seriously disrupted by attacks exploiting the vulnerabilities we identified. For example, 15% malicious nodes caused the average goodput ratio to decrease to less than 30%. (ii) Our solution is effective in mitigating the attacks; it achieves an average goodput ratio of more than 90% even when there are 30% malicious nodes conducting data dropping attacks and over 83% average goodput ratio when there are 30% malicious nodes conducting data delaying attacks. (iii) While each of the mechanisms we introduce can individually benefit the system, the solution is most effective when all the mechanisms are combined.

6 RELATED WORK

This chapter presents related work that has influenced and shaped the work contained in this thesis. We first look at research focused on modelling P2P traffic and works evaluating the effectiveness of localization. We then review previous work on attacks and defenses in Virtual Coordinate Systems. Finally, we discuss research on P2P streaming and attacks and defenses in P2P systems.

6.1 P2P Localization

Much work on modeling traffic on the Internet has been done in the context of intra-AS traffic matrix estimation [30,31]. Our work though focuses on inter-AS P2P traffic matrix estimation, of which the only related paper is [27], which we extensively discussed in Chapter 3 Sec. 3.2.

The effects of P2P systems on ASes has also been studied by Rasti et al. [115] who shows the effects of the Gnutella P2P system on the AS topology. Rather than focusing on localization, they instead study how the load on ASes due to Gnutella clients has changed over time due to the evolution of both the AS topology and the Gnutella system.

Many recent works have focused on how to implement P2P localization [19–25,40]. However, we evaluate the impact localization will have on all ASes and on their profitability.

Complementary to our work is the work by Cuevas et al. [28]. Their focus is on understanding the extent to which localization improves the performance of users and reduces the amount of P2P traffic residential ASes exchange with their providers. Similarly, Blond et al. [116] focus on how much traffic can be reduced due to localization using experiments driven by a BitTorrent crawl. In contrast, our goal is to understand the implications of localization on the global Internet, particularly, which ASes will benefit and which will lose. In addition, our analysis not only considers residential ASes but also study how localization may affect pure-transit ASes, which may not have any internal peers.

Piatek et al. [26] question the effectiveness of localization on peers performance and ISP traffic reduction. Specifically, they perform experiments showing that client-only localization policies will have limited benefits and the tracker will need to be involved to receive full benefits. They also evaluate the amount of traffic reduction possible for a crawl of one thousand torrents. In contrast, we consider a very large dataset including millions of torrents and also use realistic pricing models to understand how traffic reductions translate into impact on profit for ISPs.

6.2 Virtual Coordinate Systems

Much research has been conducted to find detection and mitigation techniques against attacks [65] in VCS.

Landmark-based defenses: Kaafar *et al.* [66] propose to model the behavior of trusted landmark nodes using a Kalman filter, this provides an outlier detection scheme by which nodes learn good behavior and can then filter out malicious updates. Their technique requires 8% of all nodes to be trusted, which could be non-trivial to obtain given a large deployment. Similarly, Saucez *et al.* [117] define a reputation based system that leverage trusted nodes and a reputation certification agent to calculate the other nodes reputation. Treeple [71], while not strictly coordinate based, provides secure latency estimation, using landmarks as vantage points for providing traceroutes on the Internet. In Treeple, landmarks perform traceroute measurements to peers, which the landmarks can then digitally sign and provide for nodes to compute the network distance themselves. As landmark-based defenses have stronger assumptions, as they require *a priori* trusted nodes, we do not compare Newton to them as Newton is a decentralized defense and does not require trusted nodes.

Decentralized defenses: Zage *et al.* [67] propose the usage of spatial and temporal properties of nodes to learn what good behavior is, then by using outlier detection detect anomalous coordinate updates. Veracity [68] uses a voting scheme to verify potentially malicious coordinate updates by using a subset of nodes. Each node maintains a verification set where several other nodes attest to whether a particular update increases their estimation error above a certain threshold, and if so, ignores it. Suspected nodes are tested based on their error to the verifying nodes; nodes with large errors are considered malicious.

Although these decentralized defenses differ in the way they secure virtual coordinate systems, they both, along with [66], suffer from the frog-boiling attack [69–71]. A few works have been proposed to defend against the frog-boiling attack. Wang *et al.* [69] proposes detecting attackers that lie about coordinates by using the PeerReview [118] accountability protocol. Since, if implemented, this approach would have higher costs than our method (i.e. bandwidth, storage for a tamper-evident log, and computation for public-key cryptography), we do not compare Newton with them. Becker *et al.* [119] propose a method for detecting frog-boiling by using a machine learning approach, where through a training data set the system learns what normal and abnormal data is. In contrast, our approach has no need to train the system and can detect abnormal behavior directly due to the applied physical laws. Furthermore, while [119] can detect attacks are occuring but not find and discard the updates that are causing it, Newton is able to do both.

6.3 P2P Streaming

Much recent work has gone on in improving the efficiency and performance of P2P streaming systems. Lui et al [120] present algorithms that find near-optimal streaming rates when nodes can only support a bounded number of children. Picconi et al [22] demonstrate that P2P live streaming systems can incorporate locality-awareness and thus be ISP-friendly. Several works [121, 122] have also focused on utilizing network coding for improving download speeds and reducing the scarcity of data. We note that works such as these are orthogonal to ours and can be incorporated with our design.

However, the security challenges in designing mesh-based streaming protocols has received little attention. Recent work [123, 124] has surveyed security issues in P2P streaming, but cite a lack of solutions in this area. The only prior work we are aware of focuses on attacks where malicious nodes pollute data sent to other nodes [101,125] or malicious nodes overload others with requests [102]. In contrast, our focus is on data availability and prevention of neighbor selection attacks.

Attacks on data availability have been considered in the context of tree-based multicast [126]. The proposed solution takes advantage of the tree structure, knowing that if a child did not receive a message then an ancestor can be traced back to that is at fault for dropping it. Meshes do not have parent-child relationships but rather nodes get data from many neighbors, so this approach cannot be applied to them. Attacks against measurement-based neighbor selection were studied in the context of tree-based streaming [127]. The proposed solution uses outlier detection to identify malicious nodes that report wrong measurement results. This approach only works with systems that employ such measurement-based adaptation.

Dealing with selfish and Byzantine behavior using game theoretic principles has been investigated in several previous works [106, 128]. Most similar to our work is Flightpath [106], a P2P streaming system that is designed to give selfish peers incentives to obey protocols and can tolerate Byzantine behavior. Unlike their work, we do not assume synchronized clocks or synchronous communication channels.

Several previous works have dealt solely with selfish users in P2P streaming. Contracts [104] develops incentives that rewards nodes by giving them higher quality playback based on how effective a node's contributions to the entire system are. Substream trading [103] applies BitTorrent's tit-for-tat mechanism to a streaming context to encourage uploading, in this context nodes commit to sending each other parts of the video stream for a period of time. Pulse [105] also applies a tit-for-tat mechanism to live P2P streaming, but also combines it with incentives for altruistic behavior.

Several schemes have been proposed to mitigate neighbor selection attacks (referred to as eclipse attacks) in the context of distributed hash tables (DHTs) [129,130]. The solutions are DHT-specific and do not apply to streaming protocols. A key aspect that distinguishes streaming protocols is the potential for feedback from the data-plane. In particular, it is possible to infer malicious behavior based on lack of data received from a neighbor. Our solutions leverage this observation resulting in significantly simpler designs.

Reputation systems have been a subject of wide interest, especially for P2P file-sharing systems. File-sharing reputation systems generally fall into two categories of purpose, incentivizing users to share files [110, 131], or thwarting file pollution [132]. Piatek et al. [131] show the feasibility of using one-hop reputations to incentivize interactions between users in BitTorrent. They take advantage of the fact that there are some users who are in many BitTorrent overlays and thus can be used as intermediaries, keeping track of long-term reputation values for others and facilitating data exchanges. While our work also uses local reputations, we differ in that our goal is mitigating malicious adversaries and not creating incentives. Also, as users usually only watch one video stream at a time, this precludes them from being in many overlays at once, making it impossible for some users to be intermediaries. Thus, streaming presents new challenges for reputation systems and has unique features that create opportunities, such as the continual downloading of data and stringent data deadlines, that we take advantage of.

7 CONCLUSION

Overlay networks provide scalable services for the distribution of content on the Internet today. As such, a large portion of traffic on the Internet is due to overlay networks. However, in the past, many P2P systems have been oblivious to network locality, thus causing an increase in the amount of traffic that must leave an ISP. P2P localization has then been proposed as a solution to contain traffic to within an ISP. Localization has since become a critical design component for overlay networks as it benefits both ISPs and end users.

However, what ISPs will really benefit economically due to localization and to what extent has not been studied previously. We conducted an extensive simulation study to understand how profits will change for ISPs once localization is deployed. To accomplish this, we proposed a new inter-AS P2P traffic model that takes into consideration the cultural and linguistic preferences that the end users in each AS have. We also proposed models for localization that allow us to understand to what degree will traffic be reduced when localization is deployed. Finally, we used realistic pricing models to calculate how reductions in traffic translate in actual profits gained. These models, together with real P2P data, allowed us to see that the benefits of localization should not be taken for granted. While some residential ISPs can received up to 90% loss reduction, other residential ISPs do not see any profit gains as they also serve as transit ISPs and thus make money on the traffic their own clients send. Furthermore, many pure transit ISPs lose a considerable amount of profit as they have to carry less traffic.

The services that provide localization can also be subject to insider attacks. Specifically, Virtual Coordinate Systems are vulnerable to attackers that lie about their coordinate values or delay measurement probes. Previous defenses have been proposed but must first learn what good behavior is and thus are subject to attacks where malicious nodes mimic good behavior. We proposed to secure VCS by introducing invariants into the system that allow us to judge whether a node is following the protocol or not. We designed three invariants that are based on Newton's three laws of motion. We found that with our defenses in place we are able to mitigate all known attacks against VCS and are able to tolerate a higher percentage of attackers than previous defenses. We also found that we are able to perform better than an unprotected VCS even in benign settings. We found that our solution is 25% more accurate and 68% more stable.

We also investigated how to protect the overlays that run on top of localization, as they too can be vulnerable to insider attackers. While mesh-based approaches have emerged as the dominant architecture for P2P streaming, the performance of these approaches under malicious participants has received little attention. We provided a taxonomy of the implicit commitments made by nodes when peering with others. We showed that when these commitments are not enforced explicitly, they can be exploited by malicious nodes to conduct attacks that degrade the data delivery service. We presented mechanisms that can enhance the resilience of mesh-based streaming against such attacks. A key part of our solution is a novel reputation scheme that combines feedback from both the control and data planes of the overlay. We evaluate our design with real-world experiments on the PlanetLab testbed and show that our design is effective. Even when there are 30% attackers, nodes receive 92% of the data with our schemes, however without our schemes they only receive 10% of the data.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[2] YouTube. http://www.youtube.com.

[3] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[4] Skype. http://www.skype.com.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the USENIX Security Symposium*, 2004.

[6] Christos Gkantsidis and Pablo Rodriguez Rodriguez. Network coding for large scale content distribution. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[7] PPLive. http://www.pplive.com.

[8] Joost. http://www.joost.com.

[9] PPMate. http://www.ppmate.com.

[10] PPStream. http://www.ppstream.com.

[11] QQLive. http://tv.qq.com.

[12] UUSee. http://www.uusee.com.

[13] Bartosz Biskupski, Raymond Cunningham, Jim Dowling, and René Meier. High-bandwidth mesh-based overlay multicast in heterogeneous environments. In *Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA)*, 2006.

[14] Nazanin Magharei and Reza Rejaie. PRIME: Peer-to-peer receiver driven mesh-based streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[15] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[16] Akamai. http://www.akamai.com.

[17] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: An overlay testbed for broad-coverage services. *SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[18] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2010.

[19] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4P: Provider portal for applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[20] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can ISPs and P2P users cooperate for improved performance? *SIGCOMM Computer Communication Review*, 37(3):29–40, July 2007.

[21] David R. Choffnes and Fabián E. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[22] Fabio Picconi and Laurent Massoulié. ISP friend or foe? Making P2P live streaming ISP-aware. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2009.

[23] Jiajun Wang, Cheng Huang, and Jin Li. On ISP-friendly rate allocation for peer-assisted VoD. In *Proceedings of the ACM International Conference on Multimedia (MM)*, 2008.

[24] Minghong Lin, John C. S. Lui, and Dah-Ming Chiu. Design and analysis of ISP-friendly file distribution protocols. In *Proceedings of Allerton Conference on Communication, Control, and Computing (Allerton)*, 2008.

[25] R. Bindal, Pei Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in BitTorrent via biased neighbor selection. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[26] Michael Piatek, Harsha V. Madhyastha, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Pitfalls for ISP-friendly P2P design. In *Proceedings of the Workshop on Hot Topics in Networks (HotNets)*, 2009.

[27] Hyunseok Chang, Sugih Jamin, Z. Morley Mao, and Walter Willinger. An empirical approach to modeling inter-AS traffic matrices. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2005.

[28] Ruben Cuevas, Nikolaos Laoutaris, Xiaoyuan Yang, Georgos Siganos, and Pablo Rodriguez. Deep Diving into BitTorrent Locality. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[29] David K. Goldenberg, Lili Qiuy, Haiyong Xie, Yang Richard Yang, and Yin Zhang. Optimizing cost and performance for multihoming. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[30] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2003.

[31] Vijayi Erramill, Mark Crovella, and Nina Taft. An independent-connection model for traffic matrices. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2006.

[32] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[33] OpenBittorrent. http://www.openbittorrent.com.

[34] Team-Cymru. IP to ASN mapping. http://www.team-cymru.org/Services/ip-to-asn.html.

[35] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, Kimberly Claffy, and George Riley. AS relationships: Inference and validation. *SIGCOMM Computer Communication Review*, 37(1):29–40, January 2007.

[36] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. IXPs: Mapped? In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2009.

[37] Jian Qiu and Lixin Gao. AS path inference by exploiting known AS paths. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2006.

[38] Oregon RouteViews. http://www.routeviews.org/.

[39] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M. Munafò, and Dario Rossi. Experiences of Internet traffic monitoring with Tstat. *IEEE Network*, 25(3), March/April 2011.

[40] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling. Traffic localization for P2P-applications: The ALTO approach. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[41] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[42] John S. Otto, Mario A. Sánchez, David R. Choffnes, Fabián E. Bustamante, and Georgos Siganos. On blind mice and the elephant: Understanding the network impact of a large distributed system. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2011.

[43] Zheng Zhang, Ming Zhang, Albert Greenberg, Y. Charlie Hu, Ratul Mahajan, and Blaine Christian. Optimizing cost and performance in online service provider networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[44] Nikolaos Laoutaris, Georgios Smaragdakis, Pablo Rodriguez, and Ravi Sundaram. Delay tolerant bulk data transfers on the Internet. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2009.

[45] Teleography Research. Telegeography international telecom trends seminar. http://www.ptc.org/ptc09/images/papers/PTC'09_TeleGeography_Slides.pdf.

[46] PeeringDB. Peering networks. https://www.peeringdb.com/.

[47] Internet Topology Collection. http://irl.cs.ucla.edu/topology/.

[48] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 1999.

[49] Dr. Peering. http://drpeering.net/AskDrPeering/blog/articles/The_Folly_of_Peering_Ratios__Intro.html.

[50] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2001.

[51] Yang-hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2000.

[52] Jonathan Ledlie, Michael Mitzenmacher, and Margo Seltzer. Wired geometric routing. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2007.

[53] Ramakrishna Gummadi and Ramesh Govindan. Reduced state routing in the Internet. In *Proceedings of the Workshop on Hot Topics in Networks (HotNets)*, 2004.

[54] James Cowling, Dan R. K. Ports, Barbara Liskov, Raluca Ada, and Popa Abhijeet Gaikwad. Census: Location-aware membership management for large-scale distributed systems. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2009.

[55] Yuval Shavitt and Tomer Tankel. Big-bang simulation for embedding network distances in euclidean space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, December 2004.

[56] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[57] T. S. Eugene Ng and Hui Zhang. A network positioning system for the Internet. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[58] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2002.

[59] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A global Internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, October 2001.

[60] Liying Tang and Mark Crovella. Virtual landmarks for the Internet. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2003.

[61] Marcelo Pias, Jon Crowcroft, Steve R. Wilbur, Timothy L. Harris, and Saleem N. Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2003.

[62] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet coordinates for distance estimation. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[63] Li-wei Lehman and Steven Lerman. A decentralized network coordinate system for robust Internet distance. In *Proceedings of the International Conference on Information Technology: New Generations (ITNG)*, 2006.

[64] Li-wei Lehman and Steven Lerman. Pcoord: Network position estimation using peer-to-peer measurements. In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA)*, 2004.

[65] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Virtual networks under attack: Disrupting Internet coordinate systems. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNext)*, 2006.

[66] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kave Salamatian, Thierry Turletti, and Walid Dabbous. Securing Internet coordinate embedding systems. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[67] David John Zage and Cristina Nita-Rotaru. On the accuracy of decentralized virtual coordinate systems in adversarial networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2007.

[68] Micah Sherr, Matt Blaze, and Boon Thau Loo. Veracity: Practical secure network coordinates via vote-based agreements. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2009.

[69] Guohui Wang and T.S. Eugene Ng. Distributed algorithms for stable and secure network coordinates. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2008.

[70] Eric Chan-Tin, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. The frog-boiling attack: Limitations of secure network coordinate systems. *ACM Transactions of Information and System Security*, 14(3):27:1–27:23, November 2011.

[71] Eric Chan-Tin and Nicholas Hopper. Accurate and provably secure latency estimation with Treeple. In *Proceedings of ISOC Symposium of Network and Distributed Systems Security (NDSS)*, 2011.

[72] Marco Mamei, Franco Zambonelli, and Letizia Leonardi. Co-fields: A physically inspired approach to distributed motion coordination. *IEEE Pervasive Computing*, 3(2):52 –61, April-June 2004.

[73] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. ANTIDOTE: Understanding and defending against poisoning of anomaly detectors. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2009.

[74] p2psim. http://pdos.csail.mit.edu/p2psim/.

[75] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *Proceedings of the ACM SIGCOMM Workshop on Internet measurment (IMW)*, 2002.

[76] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network coordinates in the wild. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[77] Yuval Shavitt and Tomer Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2004.

[78] Cristian Lumezanu and Neil Spring. Measurement manipulation and space selection in network coordinates. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2008.

[79] Yongquan Fu and Yijie Wang. Hyperspring: Accurate and stable latency estimation in the hyperbolic space. In *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*, 2009.

[80] Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[81] Yang-hua Chu, Aditya Ganjam, T. S. Eugene Ng, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang. Early experience with an Internet broadcast system based on overlay multicast. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[82] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, John C. S. Lui, and Cheng Huang. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[83] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[84] Vinay Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, and Alexander Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2005.

[85] Vidhyashankar Venkataraman, Kaouru Yoshida, and Paul Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[86] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shng Peter Yum. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[87] Dejan Kostić, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[88] SOPCast. http://www.sopcast.org/.

[89] Pdbox. http://www.pdbox.co.kr.

[90] StreamerOne. http://www.streamerone.com.

[91] TVUnetworks. http://www.tvunetworks.com.

[92] VGO. http://vgo.21cn.com.

[93] Zattoo. http://zattoo.com.

[94] Veetle. http://www.veetle.com, 2010.

[95] Venkata N. Padmanabhan, Helen J. Wang, and Philip A. Chou. Resilient peer-to-peer streaming. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2003.

[96] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, Jr. Overcast: Reliable multicasting with on overlay network. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[97] Dongni Ren, Y.-T. Hillman Li, and S.-H. Gary Chan. On reducing mesh delay for peer-to-peer live streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[98] Feng Wang, Jiangchuan Liu, and Yongqiang Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[99] Jiangchuan Liu, Sanjay G. Rao, Bo Li, and Hui Zhang. Opportunities and challenges of peer-to-peer internet video broadcast. In *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[100] Jeff Seibert, David Zage, Sonia Fahmy, and Cristina Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. In *IEEE Conference on Local Computer Networks (LCN)*, 2008.

[101] Prithula Dhungel, Xiaojun Hei, Keith Ross, and Nitesh Saxena. The pollution attack in P2P live video streaming: Measurement results and defenses. In *Proceedings of the SIGCOMM P2P-TV Workshop*, 2007.

[102] Maya Haridasan and Robbert van Renesse. Defense against intrusion in a live streaming multicast system. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2006.

[103] Zhengye Liu, Yanming Shen, Keith Ross, Shivendra. S. Panwar, and Yao Wang. Substream trading: Towards an open P2P live streaming system. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2008.

[104] Michael Piatek, Arvind Krishnamurthy, Arun Venkataramani, Richard Yang, David Zhang, and Alexander Jaffe. Contracts: Practical contribution incentives for P2P live streaming. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[105] Fabio Pianese, Student Member, Diego Perino, Joaqun Keller, and Ernst W. Biersack. Pulse: An adaptive, incentive-based, unstructured P2P live streaming system. *IEEE Transactions on Multimedia*, 9(8):1645–1660, December 2007.

[106] Harry C. Li, Allen Clement, Mirco Marchetti, Manos Kapritsos, Luke Robison, Lorenzo Alvisi, and Mike Dahlin. Flightpath: Obedience vs. choice in cooperative services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2008.

[107] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 42(1):1:1–1:31, December 2009.

[108] John R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2002.

[109] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. *IEEE/ACM Transactions on Networking*, 18(3):885–898, June 2010.

[110] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2003.

[111] Tyson Condie, Varun Kacholia, Sriram Sankararaman, Joseph M. Hellerstein, and Petros Maniatis. Induced churn as shelter from routing table poisoning. In *Proceedings of ISOC Symposium of Network and Distributed Systems Security (NDSS)*, 2006.

[112] Ashwin R. Bharambe, Sanjay G. Rao, Venkata N. Padmanabhan, Srinivasan Seshan, and Hui Zhang. The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2005.

[113] Kevin Almeroth and Mostafa Ammar. Characterization of mbone session dynamics: Developing and applying a measurement tool. Technical Report GIT-CC-95-22, Georgia Institute of Technology, 1995.

[114] Bo Li, Susu Xie, Gabriel Y. Keung, Jiangchuan Liu, Ion Stoica, Hui Zhang, and Xinyan Zhang. An empirical study of the Coolstreaming+ system. *IEEE Journal on Selected Areas in Communications*, 25(9):1627–1639, December 2007.

[115] Amir Hassan Rasti, Reza Rejaie, and Walter Willinger. Characterizing the global impact of P2P overlays on the AS-level underlay. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)*, 2010.

[116] Stevens Le Blond, Arnaud Legout, and Walid Dabbous. Pushing BitTorrent locality to the limit. *Computer Networks*, 55(3):541–557, February 2011.

[117] Damien Saucez, Benoit Donnet, and Olivier Bonaventure. A reputation-based approach for securing Vivaldi embedding system. In *Proceedings of the Conference on Dependable and Adaptable Networks and Services*, 2007.

[118] Andreas Haeberlen, Petr Kuznetsov, and Peter Druschel. PeerReview: Practical accountability for distributed systems. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2007.

[119] Sheila Becker, Jeff Seibert, Cristina Nita-Rotaru, and Radu State. Securing application-level topology estimation networks: Facing the frog-boiling attack. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2011.

[120] Shao Liu, Minghua Chen, Sudipta Sengupta, Mung Chiang, Jin Li, and Phil A. Chou. P2P streaming capacity under node degree bound. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2010.

[121] Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. UUSee: Large-scale operational on-demand streaming with random network coding. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[122] Anh Tuan Nguyen, Baochun Li, and Frank Eliassen. Chameleon: Adaptive peer-to-peer streaming with network coding. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[123] Jan Seedorf. Security issues for P2P-based voice- and video-streaming applications. In *Proceedings of Open Research Problems in Network Security (iNetSec)*, 2009.

[124] Gabriela Gheorghe, Renato Lo Cigno, and Alberto Montresor. Security and privacy issues in P2P streaming systems: A survey. *Peer-to-Peer Networking and Applications*, 2010.

[125] Qiyan Wang, Klara Nahrstedt Long Vu, and Himanshu Khurana. Identifying malicious nodes in network-coding-based peer-to-peer streaming networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[126] Liang Xie and Sencun Zhu. Message dropping attacks in overlay networks: Attack detection and attacker identification. *ACM Transactions of Information and System Security*, 11(3):15:1–15:30, March 2008.

[127] AAron Walters, David Zage, and Cristina Nita-Rotaru. Mitigating attacks against measurement-based adaptation mechanisms in unstructured multicast overlay networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[128] Idit Keidar, Roie Melamed, and Ariel Orda. Equicast: Scalable multicast with selfish users. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2006.

[129] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[130] Atul Singh, Tsuen-Wan Ngan, Peter Druschel, and Dan S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[131] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[132] Kevin Walsh and Emin Gün Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2006.

VITA

VITA

Jeffrey Seibert was born in Evansville, Indiana. He received his Bachelor of Science in Computer Science and Mathematics from Purdue University in 2006. He received his Ph.D. in Computer Science in 2012 from Purdue University. During his time at Purdue, he was a member of the Dependable and Secure Distributed Systems Lab and was affiliated with the Center for Education and Research in Information Assurance and Security (CERIAS). His research focused on designing and building distributed systems that can tolerate malicious insiders and can continue to perform well even when under attack. He also conducted research in developing automated approaches to finding attacks in real implementations of distributed systems.