COORDINATE-BASED ROUTING FOR HIGH PERFORMANCE ANONYMITY

Micah Sherr

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2009

Matt Blaze and Boon Thau Loo
Supervisors of Dissertation

Jianbo Shi
Graduate Group Chairperson

Acknowledgments

Many people contributed ideas, offered suggestions and criticisms, and provided insights that significantly helped develop this dissertation. I apologize profusely for any omissions I may have made below. I am profoundly grateful for the guidance, friendship, and support of all my advisers, colleagues, friends, and family that made this dissertation possible.

After graduating from the University of Pennsylvania (for the first time) in May 2000, I joined Scient, one of the fastest growing (and subsequently fastest shrinking) IT consulting firms in history. After Scient went belly-up[1] in 2001, I moved to New York City to be closer to my girlfriend, Lisa, and took a job as a systems programmer at Columbia University (an institution I reasoned was unlikely to go bankrupt). My bosses at Columbia – Suleman Hamid and Maria Mosca – as well as Cosmin Zamfirescu, a good friend and colleague, encouraged me to take advantage of the University and enroll in graduate computer science classes. One of those courses, *Advanced Internet Services* taught by Dr. Henning Schulzrinne, spawned my interest in systems research. An independent research course with Dr. Schulzrinne the following semester furthered that interest. Finally, weekly lunches with Matt Burnside, a friend and former roommate who came to Columbia for his PhD in computer science, convinced me to apply to PhD programs.

I started at Penn (for the second time) in Fall 2003. Moving to Philadelphia

[1] I contend through no fault of my own

meant that I was also moving away from Lisa, but SEPTA and New Jersey Transit trains meant that we could see each other often. My first adviser at Penn, Professor Carl Gunter, gave me a much needed crash course on being an effective graduate student. Carl, along with his graduate students – Aaron Evans, Alwyn Goodloe, Michael May, Michael McDougall, and Kaijun Tan – taught me the fundamentals of security research that served me well for the remainder of my graduate student life. Even after he left the University to lead the Security Lab at the University of Illinois at Urbana-Champaign, Carl funded me throughout my first year, enabling me to continue at Penn.

I met my second adviser, Dr. Matt Blaze, when he came to Penn in Spring 2004. Taking on an enthusiastic but inexperienced advisee for five or more years is inarguably a risky proposition for a professor, and it is one that I am very grateful that Matt was willing to make. I could devote several chapter-length sections to enumerating the ways in which Matt helped guide me (mostly unscathed) through graduate school. In the interests of brevity and for the sake of completing this thesis while my hairline is still intact, I'll instead describe two memorable experiences that I believe are unique to being one of Matt's students:

In Fall 2005, Matt, fellow advisees Sandy Clark and Eric Cronin, and I found serious vulnerabilities in deployed law enforcement wiretap systems [98]. Usually, academic research in computer science is intellectually stimulating but rarely makes an impact outside of niche academic communities. In this case, however, our research led us to a face-to-face meeting with the FBI. Fortunately, our work was well received and none of us were jailed.

As a second example, I participated in a team Matt led in California's Top-to-Bottom Review of electronic voting machines. Although the California study required me to spend daylight hours in a windowless and underventilated office and my nights in a dorm room on the University of California, Berkeley campus, working with Matt and the other teammates on such an important and politically relevant study was the highlight of my academic career.

I have learned an innumerable amount from Matt. He is a mentor in the truest sense of the word.

I owe a tremendous amount to my co-adviser, Dr. Boon Thau Loo. In Spring 2007 I took Boon's *Networking Meets Databases* seminar course. The final project for that class, under Boon's tutelage, eventually morphed into this thesis. For the past two years, I have looked forward to weekly meetings in Boon's office in which he would usually tell me that the previous week's work was good, but that it could be substantially improved by doing $\mathcal{X}$, where $\mathcal{X}$ is invariably a brilliant idea that could not have come from anybody else. Boon's ability to poke holes in otherwise convincing arguments, his encyclopedic knowledge of scientific literature, and his willingness to dive headfirst into gritty details have been invaluable to me. I am deeply indebted to him.

My colleagues along the way have challenged, motivated, and inspired me, alleviated my stress with laughter, and made my tenure as a graduate student the best job (to use the word loosely) I have ever had. In particular, I would like to acknowledge "the team": Adam Aviv, Sandy Clark, Eric Cronin, and Gaurav Shah. Working long hours with the same four people for six years can sometimes be challenging. This was most definitely not the case with Adam, Sandy, Eric, and Gaurav. I will miss their intelligence, humor, and quick-wittedness after we depart from Penn.

Also at Penn, Madhukar Anand, Colin Blundell, and TJ Green have been invaluable friends and colleagues. Pavol Černý, Nate Foster, and Dimitrios Vytiniotis deserve special mention for putting up with me as a housemate. After I relocated to faraway Morristown, NJ, I sometimes slept in our lab whenever I needed to stay overnight in Philadelphia. After hearing about one particular incident in which the air conditioning was set too high in the lab, forcing me to cuddle with my laptop for warmth, Nate and his roommate, Chris Thornton, graciously opened up their (significantly warmer) house to me.

In the summer of 2006, I interned at Intel Research in Santa Clara, CA. Eve Schooler, the project lead of Intel's *Distributed Detection and Inference* (DDI) team, was a fantastic mentor. Jaideep Chandrashekar, my direct supervisor and friend, challenged me at work and entertained me afterwords by introducing me to the many fine eating establishments in Silicon Valley. Other Intel team members – John Agosta, Denver Dash, and Carl Livadas – offered insights from various disciplines within computer science, and were thankfully patient with me as I attempted to expand the breadth of my knowledge. Carlos Diuk, a friend, fellow intern, and cohabiter of Intel cubicle 6-L-23-μ is the only person I have ever met who is able and willing to rearrange burning coals on a grill with his bare hands. He also cooks a fantastic steak.

The California and Ohio voting efforts involved many people. During the California Top-to-Bottom Review, I worked with an extremely talented group, composed of Matt Blaze, Arel Cordero, Sophie Engle, Chris Karlof, Naveen Sastry, Till Stegers, and Ka-Ping Yee. Professor David Wagner served as the project lead. The Penn component of the Ohio EVEREST project consisted of Matt Blaze, Adam Aviv, Sandy Clark, Eric Cronin, Gaurav Shah, and myself.

During the process of writing this dissertation, I received enormously helpful feedback from my thesis committee. I thank Professors Jonathan Smith, Roch Guerin, David Wagner, and Steve Zdancewic for their many helpful suggestions and comments. Of course, this work could not have been possible without the help of my co-advisors, Matt Blaze and Boon Thau Loo.

No CIS graduate student completes a thesis without significant support from the department's staff members. In particular, Mike Felker, the amazingly multitasking Graduate Coordinator, helped me successfully navigate Penn's numerous bureaucracies and never appeared exasperated over the occasional tardiness of my paperwork.

I owe a great deal of thanks to my friends who reside outside of the computer science ecosystem. In particular, Matt Davids, Billie-Jo Warrick McIntire, Stephen

Rabin, Mary Kay Taylor, Reid Weisbord, and Ben Zotto listened to my gripes about the life of a doctoral student and offered sage advice and needed distractions.

My mother, Professor Mary-Lou Breitborde, is likely responsible for my interest in academia. My very first adviser, she is always my tutor in life. She and her partner, Robert Scheier, have encouraged me throughout this six year process. I am grateful for their advice and for keeping their door open whenever I needed to return home.

My father, Alan, inspired me to become an engineer – a position that, in my youth, I was convinced involved piloting a locomotive. His love for science and his dedication to using knowledge to improve humanity inspires me to do the same. My father along with my stepmother, Laura Bjorklund, have been an invaluable source of wisdom. I promise them that once this dissertation earns me a job, I will give them their car back.

My brothers, Reuben and Jonas, keep me balanced. They remind me that life is much more than long sequences of zeroes and ones.

My maternal grandparents, Abraham and Sylvia Breitborde, passed away before I started my graduate studies. I think about them often. Memories of happy holidays with them can brighten the bleakest of days. My paternal grandmother, Charlotte, is a source of inspiration. Her sense of humor always brings a smile. Her husband and my grandfather, Simon, died in 2005. I deeply miss his wisdom. I think he would have been glad to see me finish my studies.

Finally and most significantly, I owe too much to express in words to Lisa, my girlfriend from college who accepted with little coaxing my wacky plan to return to school for at least another half-decade. When I left for Philadelphia, she remained in New York. The constant drudging back and forth on New Jersey Transit proved too much for us, so she moved to Philadelphia so that we could be together. Her patience, love, and partnership made the journey that culminated in this dissertation an easy one. We married in August 2007. This thesis is for us and for our future.

ABSTRACT

COORDINATE-BASED ROUTING FOR HIGH PERFORMANCE ANONYMITY

Micah Sherr

Matt Blaze and Boon Thau Loo

Today's Internet routing protocols, while arguably robust and efficient, are not designed to support anonymous communication. Internet packets must include accurate destination addresses to be routable and truthful source information to achieve reliability. While there have been several attempts at providing anonymity with the use of application-level overlay networks, these solutions focus almost exclusively on maximizing anonymity, typically at the expense of performance.

This dissertation shows that it is both possible and practical to design, secure, and scale decentralized overlay networks for high performance anonymous routing. Our techniques utilize virtual coordinate systems that embed link information (for example, latency, jitter, and loss) in n-dimensional coordinate planes. Such coordinate systems enable nodes to estimate pairwise network metrics between remote peers without requiring direct measurements. We introduce methods for scalably disseminating coordinate information as well as security mechanisms for enforcing truthful coordinate advertisements. By allowing nodes to estimate the end-to-end performance of possible routes, our overlay routing infrastructure empowers applications to intelligently select high performing anonymous paths.

Unlike existing anonymity systems that depend on central authorities or directories, our *coordinate routing system* does not rely on *a priori* trusted nodes or third-party authorities. This lack of centralization enables our system to scale to potentially millions of nodes and offer anonymity that does not depend on the trustworthiness of select nodes or services. Moreover, the ability to estimate the end-to-end performance of potential anonymous paths and prune likely underperforming routes permits the anonymization of high bandwidth and low latency network services (for example, voice-over-IP, streaming video multicast, etc.) whose communication requirements have previously been considered too restrictive for anonymity networks.

Contents

Acknowledgments **iii**

1 Introduction **1**

1.1 Problems with Existing Anonymity Systems 5

1.2 Achieving High Performance Anonymity 7

1.3 Research Questions 9

1.4 Contributions 10

1.5 Organization 11

2 Related Work **13**

2.1 Terminology 13

2.2 Anonymity Systems 15

2.2.1 Crowds 16

2.2.2 Onion Routing and Tor 18

2.3 Path Selection 20

2.4 Attacks on Anonymity 22

2.4.1 Counting Attacks 23

2.4.2 Low-Latency Attacks 24

2.4.3 Predecessor Attacks 25

2.4.4 Timing Attacks 25

2.4.5 Path Selection Attacks 27

3 Link-Based Path Selection 30

3.1 Weighted Path Selection 30

3.2 Assumptions and Limitations 32

3.3 The Case for Link-based Selection 34

3.3.1 Bandwidth 35

3.3.2 Non-bandwidth Metrics 37

4 Practical Link-Based Path Selection 38

4.1 Background: Vivaldi Coordinate System 39

4.2 Metrics 40

4.3 Performance Impact of Coordinate Systems 41

4.4 Coordinate System Security 45

4.5 Locating the Responder 45

4.5.1 Closest Relay Services 46

4.5.2 AS Path 46

4.5.3 Disconnected Endpoints 47

5 Anonymity Analysis 49

5.1 Attacker Model 49

5.2 Node Prevalence: A New Metric for Anonymous Path Selection . . . 50

5.2.1 Node Prevalence for Link-Based Relay Selection 51

5.2.2 Node Prevalence for Node-Based Relay Selection 52

5.3 Attack Strategies 53

5.3.1 `BestLinks`: Compromising Attractive Links 54

5.3.2 `MedianDist`: Compromising Nodes with Shortest Median Distances 56

5.3.3 `Prevalence`: Compromising Nodes with Greatest Node Prevalence 57

5.3.4 `Confirmation:` Determining whether Alice is Communicating with Bob 58
5.3.5 `Cluster:` Joining the Network with a Cluster of Nodes 59
5.4 Anonymity Benefits of WEIGHTED-DETACHED 61
5.4.1 Resilience to `Confirmation` Attacks 62
5.4.2 Resilience to `MedianDist` and `Prevalence` Attacks 63
5.4.3 Preventing the Predecessor Attack 64
5.5 The Impact of Coordinate Systems on Anonymity 65

6 Application-Aware Anonymity (A^3) 69
6.1 Design Goals, Assumptions, and Limitations 70
6.2 Performance Advantages of a Fully Distributed Architecture 72
6.3 System Architecture 73
6.3.1 Distributed Directory Service 75
6.3.2 Embedded Coordinate System 78
6.3.3 Anonymous Lookup Tunnels 79
6.3.4 Routing Engine 82
6.4 Implementation and Evaluation 83
6.4.1 Implementation 83
6.4.2 Simulation Results 84
6.4.3 PlanetLab Evaluation 89
6.5 Summary 92

7 Securing Coordinate Systems with Veracity 93
7.1 Comparison to Other Coordinate Protection Systems 96
7.2 Attacker Model 97
7.3 Metrics 98
7.4 Overview of Veracity 98
7.5 Veracity Verification Protocols 100

7.5.1 VSet Construction 100
7.5.2 Locating and Updating VSet Members 102
7.5.3 Publisher Coordinate Verification 102
7.5.4 Tuning VSet Parameters 103
7.5.5 Candidate Coordinate Verification 104
7.6 Distributed Directory Services 106
7.7 Implementation and Evaluation 106
7.7.1 Experimental Setup 107
7.7.2 Veracity in the Absence of Attacks 108
7.7.3 Disorder Attacks 111
7.7.4 Repulsion and Isolation Attacks 118
7.7.5 PlanetLab Results 119
7.8 Summary 121

8 Contour: Coordinate-Based Routing for Performance 123
8.1 Background: Detour Routing 123
8.2 Motivation: Trace-driven Analysis 125
8.3 System Architecture 128
8.3.1 Overview 128
8.3.2 Relay Selection Strategies 130
8.4 Evaluation 132
8.4.1 Trace-driven Simulation 132
8.4.2 PlanetLab Deployment 135
8.5 Summary 137

9 Conclusion 139

List of Tables

3.1 Link concatenation operators for anonymous paths 31

3.2 Network datasets used to evaluate link-based relay selection 35

6.1 Regression analysis of PlanetLab bandwidth requirements 90

7.1 Properties of latency datasets 104

7.2 Veracity's relative system error ratios 117

8.1 Triangle inequality violations for various latency datasets 125

List of Figures

1.1 A sample anonymous path 4

1.2 A map of the Tor network 5

1.3 Bandwidth speeds on the Tor network 6

2.1 A three layered onion 19

2.2 Example counting attack 23

3.1 E2e available bandwidth using the $\texttt{S}^3\texttt{-BW}$ dataset 36

3.2 E2e path latencies using the `King` dataset 36

3.3 E2e jitter using the `PL-Jitter` dataset 36

3.4 E2e AS traversals using the `PL-ASes` dataset 36

4.1 Coordinate system embedding errors for latency, jitter, and AS count metrics 42

4.2 Resultant path latencies of actual- and coordinate-based path selection 43

4.3 Resultant path jitter of actual- and coordinate-based path selection . 43

4.4 Resultant AS traversals of actual- and coordinate-based path selection 44

4.5 Path performance of the WEIGHTED-DETACHED relay selection strategy 47

5.1 Node Prevalence for Link-Based Relay Selection 52

5.2 Node Prevalence for Node-Based Relay Selection 53

5.3 Effectiveness of the `BestLinks` attack strategy 54

5.4 Effectiveness of the `BestNodes` attack strategy 55

5.5 Effectiveness of the `MedianDist` attack strategy 56
5.6 Effectiveness of the `Prevalence` attack strategy 58
5.7 Effectiveness of the `Confirmation` attack strategy 59
5.8 Effectiveness of the `Cluster` attack strategy 60
5.9 Effectiveness of the `Confirmation` attack strategy against Weighted-Detached 62
5.10 Effectiveness of the `MedianDist` attack strategy against Weighted-Detached 63
5.11 Effectiveness of the `Prevalence` attack strategy against Weighted-Detached 64
5.12 Maximum node prevalences for actual and coordinate-based distances 66
5.13 Effectiveness of the `MedianDist` attack strategy when the initiator uses coordinate-based distance estimates 67

6.1 A^3 system architecture 74
6.2 Distribution of A^3 path estimation errors for the `King`, `PL-ASes`, and `PL-Jitter` datasets 86
6.3 A^3 path performance for the `King`, `PL-ASes`, and `PL-Jitter` datasets 87
6.4 Locations of A^3 PlanetLab nodes 88
6.5 A^3 bandwidth utilization on PlanetLab 89
6.6 Predicted bandwidth utilization for large deployments 89
6.7 A^3 performance on PlanetLab 91

7.1 Publisher coordinate verification 101
7.2 Candidate coordinate verification 105
7.3 CDFs for median error ratios 108
7.4 Veracity system error ratio after new nodes join network 109
7.5 System error ratio for Vivaldi and Veracity under various degrees of churn 110

7.6 Veracity's effectiveness against naïve attack 112

7.7 Veracity's effectiveness against coordinated attackers 115

7.8 CDF of median error ratios using one of two protection schemes . . . 116

7.9 Veracity's resilience to repulsion and isolation attacks 118

7.10 PlanetLab bandwidth 120

7.11 Extrapolated bandwidth (KBps) for large networks 120

7.12 System error ratio achieved on PlanetLab 121

8.1 RTT improvements obtained via an optimal single-hop detour 126

8.2 RTT improvements obtained via single-hop detours using only incomplete network knowledge 127

8.3 Contour architecture 129

8.4 The distribution of median error ratios of the `Meridian` dataset . . . 133

8.5 Percent reductions in RTT achieved by Contour 134

8.6 The relationship between RTTs achieved by the direct path and by Contour 135

8.7 Cumulative distribution of percent decreases in RTT achieved by Contour on PlanetLab 136

Chapter 1

Introduction

Cryptographers have been inventing means to protect the contents of confidential communiques for thousands of years [46]. In that time, well-studied and robust techniques have been developed to securely share secrets, even across large, decentralized, and untrustworthy networks [24]. To prevent unauthorized parties from learning the contents of her communications, a principal (Alice) encrypts her messages prior to transmission. The network routes Alice's messages on a best-effort basis towards their intended recipient (Bob). Nodes along the path from Alice to Bob routinely inspect and duplicate Alice's packets (a requirement of any packet-switched network), yet even the most curious eavesdropper cannot discern meaningful content from Alice's messages without the decryption key.

Although Alice and Bob may confidently employ cryptographic solutions to preserve the secrecy of their messages across the network, the fact that the two parties are communicating is easily discernible. The information content (i.e., payload) of Alice's messages is secure, but the endpoints of the communication must be exposed to permit the messages to be propagated through the network. In-transit capture or *post facto* analysis of Alice's packets reveal the two participants as well as the time and duration of their communication.

Interestingly, the design of the Internet seems incompatible with anonymous communication. IP packets contain source and destination addresses that identify the communicating parties.[1] If Bob's address is hidden or absent, unicast message delivery is impossible. (Although broadcast may be used to connect the two parties, broadcast scales poorly beyond the local subnet and is infeasible as an Internet-wide anonymity technique.) Alice may attempt to shield her identity by falsifying the source address in transmitted packet headers. However, since many important networking features (e.g., flow control, reliability, error handling, network diagnostics, etc.) depend on bidirectional communication [75, 76], doing so significantly impairs Alice's ability to communicate.

There are many reasons, both legitimate and illicit, why Alice and Bob may wish to hide the fact that they are communicating. Uses for anonymity on the Internet include (but are not limited to) the following:

- **Protection from censorship/repressive governments.** Anonymity allows constituents of repressive regimes to access and produce news media, blogs, support groups, etc.

- **Whistle-blower protection.** Anonymity affords whistle-blowers the ability to publicize perceived wrongdoings without fear of retribution.

- **Unmonitored access to health and medical information.** Embarrassment or fear of discrimination may be impediments to accessing online health and medical resources. Anonymity provides a mechanism for individuals to freely access medical information, as well as anonymously participate in online support groups and

[1] Note that while IP addresses are generally sufficient for message delivery, they do not necessarily constitute a reliable mapping between logical communication endpoints and the conversing participants. There is a wide detachment between packets intercepted on the wire and the individuals who send and receive them [17, 18]. For example, proxy services, virtual private networks (VPNs), dynamic IP addresses, mobile IP, and network address translation (NAT) all obfuscate the communicating parties. Most literature on anonymity ignores this potentially harder problem of resolving identity. For simplicity, we adopt a similar convention and equate network addresses with human identity.

discussion boards.

- **Privacy protection.** Anonymity prevents data brokers from associating online behavior with a particular identity.

- **Blind auctions.** To ensure fair auctions, online auction houses often mask the identity of bidders until the conclusion of bidding. This is known as a *blind auction*. Bidders can either trust the bidding site to provide anonymity to all parties, or they can themselves employ anonymity techniques to ensure that their identity is not revealed until the bidding has concluded [105].

- **Isotropism.** Anonymity networks can be used to construct specialized broadcast media called *isotropic channels* in which eavesdroppers cannot discern the sender of any particular transmission. Isotropic channels can be used to share secrets with confidentiality that asymptotically reaches perfect secrecy [43, 3, 97].

- **Anonymity in Non-Internet Systems and Networks.** The Internet is becoming increasingly connected to traditionally isolated networks (for example, the PSTN and SMS networks). The Internet may be leveraged to achieve a level of anonymity not previously attainable on these disparate systems. For example, anonymous emailers can be used to send anonymous SMS messages [32], and voice-over-IP services permit caller-ID spoofing [77, 108]. Additionally, anonymous techniques designed for use on computer networks are sometimes applicable to offline technologies [72].

- **Illicit and illegal activities.** The ability to communicate anonymously reduces the fear of discovery for both illicit and illegal online activities. The latter category includes copyright infringement and other crimes against intellectual property, access to illegal materials, communication supporting organized crime and/or terrorism, and other subversive or unlawful activities.

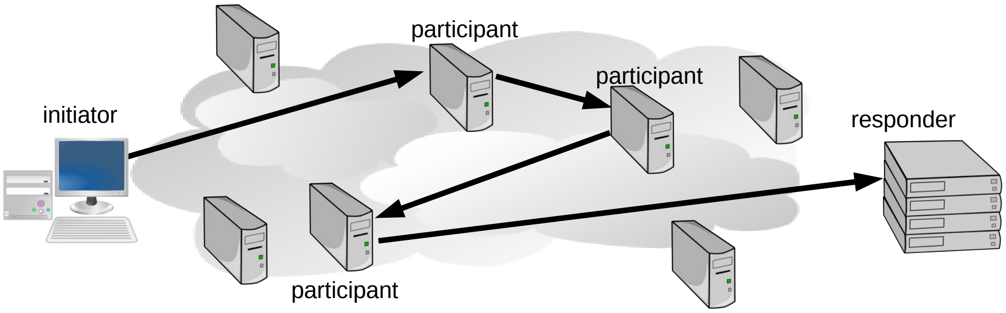


Figure 1.1: A sample three-participant anonymous path.

- **Law enforcement.** Law enforcement agencies may utilize anonymity networks for online sting operations. Anonymity hides the identity of law enforcement officials who infiltrate online communities where crimes are committed.

A myriad of techniques have been proposed for achieving anonymity on the Internet [14, 28, 84, 83, 61, 101, 37, 99]. Such approaches typically rely on overlay networks in which cryptography is used to route messages towards their intended recipients without revealing either the content or destination of messages to eavesdroppers. These techniques operate at the application-layer and typically do not require the cooperation of Internet service providers (ISPs) or autonomous systems (ASes).

In keeping with standard terminology [116], we will refer to the originator of the communication as the *initiator* (Alice). The end-to-end communication occurs between the initiator and a *responder* (Bob), the latter of whom need not be aware of the anonymity protocol.

Internet anonymity protocols typically operate as follows. The initiator sends her messages towards the responder through a series of randomly selected anonymizing *proxies* (also called *relays*). The chosen relays are known as *participants.* Participants serve as intermediaries between the communication endpoints. Cryptography prevents an eavesdropper who intercepts a message in transit from one participant to

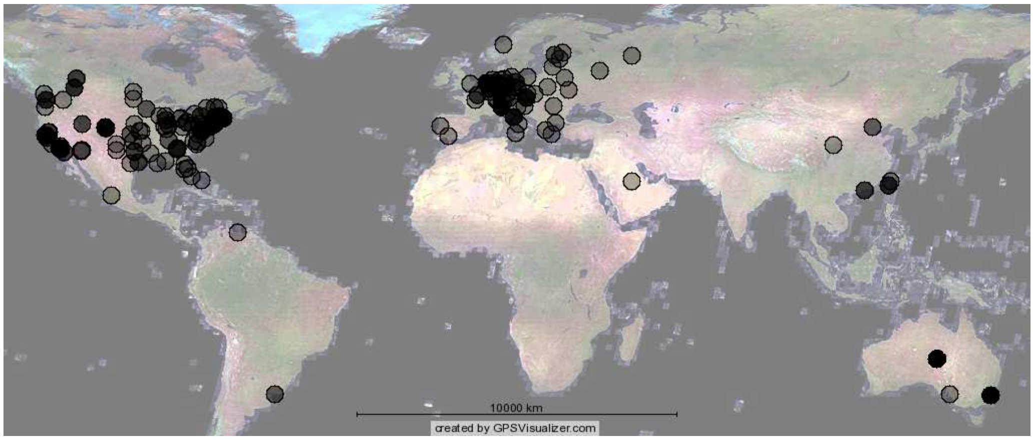


Figure 1.2: A map of the Tor network.

another from discerning the identities of the initiator and/or the responder. The last participant in an *anonymous path* (also called an *anonymous circuit*) relays the message to the responder. Reply traffic generated by the responder typically traverses the reverse path towards the initiator. An example three-participant anonymous path is depicted in Figure 1.1.

1.1 Problems with Existing Anonymity Systems

One of the must studied anonymity services is the Tor network [28], a collection of approximately 1500 quasi-permanent *onion routers* located mostly in the United States and Europe (see Figure 1.2). Tor is loosely based on the concept of *onion routing* [82] in which session keys are generated by the initiator and distributed to participants via a multiply encrypted *onion*. The onion allows each participant to decipher only the previous and next hop along the anonymous path, preventing (for instance) a rogue participant from discerning the identities of the initiator and/or responder. Importantly, Tor is a *source-routing* technique – that is, the initiator selects the participants that form the anonymous path. Initiators discover participants by downloading a copy of a *directory* maintained by centralized servers. Tor

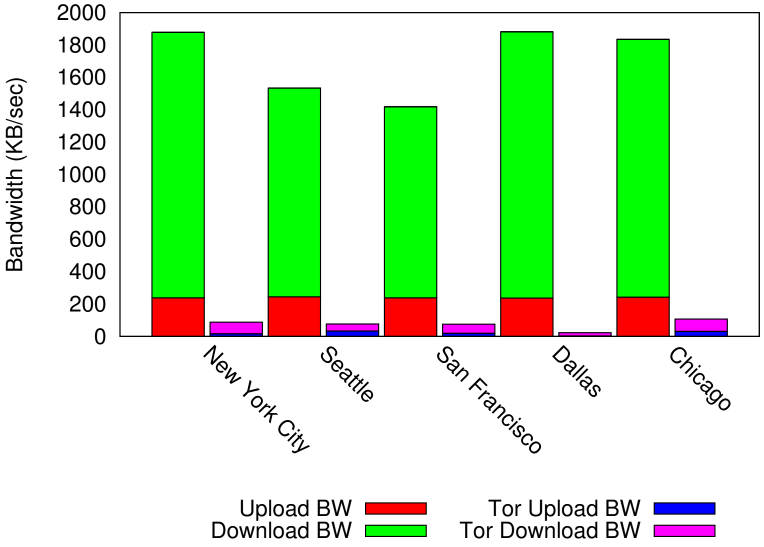


Figure 1.3: Bandwidth speeds on the Tor network, as measured using the SpeakEasy bandwidth tester from a source in New Jersey.

and other anonymity services are discussed in depth in the next chapter. Since Tor has received significant study from the academic community and because it operates today as a live anonymity network servicing hundreds of thousands of users [60], it will serve as the basis for comparison for the work presented in this dissertation.

Unfortunately, Tor often produces anonymous paths with poor performance. For example, Figure 1.3 shows the upstream and downstream bandwidths obtained using the SpeakEasy bandwidth tester [104] from a source in New Jersey to five web servers located throughout the United States using standard IP networking and the Tor network. Anonymizing connections with Tor results in an approximately tenfold decrease in both upstream and downstream bandwidth.

We postulate that several factors contribute to Tor's poor performance: First, the Tor network is comprised of just 1500 anonymizing routers, hence the number of clients accessing the network greatly exceeds the number of available relays, causing significant congestion in the network. Second, the Tor path selection algorithm selects participants according to a weighted probability distribution in which nodes are ranked according to *self-reported* available bandwidths [27, 65]. These bandwidths are not necessarily accurate (in fact, adversaries may falsely advertise high bandwidths to attract traffic [8]). Finally, Tor defines performance only in terms of bandwidth. However, e2e path performance is better characterized using multiple metrics (e.g., bandwidth availability, bandwidth capacity, latency, jitter, loss-rates, etc.). Although bandwidth is arguably a *node characteristic* [2, 50], latency, jitter, and loss-rate are *link characteristics* and require a pair of nodes to be meaningful. These latter link-based performance metrics are incompatible with the existing Tor implementation.

1.2 Achieving High Performance Anonymity

In this dissertation, we show that it is both possible and practical to design, secure, and scale decentralized overlay networks for high performance anonymous routing. Unlike existing anonymity systems in which path selection is based on self-reported node characteristics (e.g., bandwidth [28, 27]), our proposed *coordinate routing system* utilizes *link characteristics* to accurately estimate the e2e performance properties of potential anonymous paths. Our path selection algorithm considers multiple performance metrics (e.g., latency, jitter, loss, number of AS crossings, etc.) and produces high performance anonymous paths using probability distributions over candidate paths' cost estimations.

We argue that there exist several applications that can benefit from anonymity with tunable performance and security properties. At one extreme, an anonymous email system may require very strong anonymity while imposing no constraints on bandwidth or latency. Here, maximizing network diversity to reduce the feasibility of omniscient eavesdropping is paramount. At the other extreme, a unidirectional video broadcast application may select a path of participants that maximizes bandwidth and minimizes jitter with no demands for end-to-end latency, and be willing to tolerate anonymity that provides only probable innocence [83]. In between extremes, a low-bandwidth VoIP application may desire paths that minimize e2e latency, while achieving strong anonymity that ensures that paths traverse multiple ASes.

A (non-comprehensive) list of applications that may benefit from high performance anonymity include:

- **Voice-over-IP (VoIP):** VoIP applications require low latency, jitter, and loss for seamless and uninterrupted audio. The bandwidth requirements of VoIP are modest.

- **Internet TV (IPTV):** Multicast IPTV streams require high bandwidth connections with little jitter, but can tolerate high latency and modest loss.

- **Video conferencing:** Video conferencing requires high bandwidth, low latency, and low jitter. Some amount of loss may be tolerated.

- **Gaming:** Online multiplayer games typically require very low latency, but do not impose significant bandwidth requirements.

- **Large file transfers:** Transferring large files requires high bandwidth, but makes no demands on latency or jitter.

To estimate link characteristics without requiring pairwise measurements, our techniques utilize virtual coordinate systems that embed link information in n-dimensional coordinate planes [19, 16, 66, 93]. Such coordinate systems enable nodes to estimate pairwise network metrics between remote peers without requiring direct measurements.

Our use of coordinate systems is novel in two respects. First, we are the first to demonstrate that several link characteristics other than latency (e.g., jitter, loss, number of AS crossings, etc.) can be accurately embedded in a coordinate system. Second, we utilize coordinates to select high performance paths, defining the necessary aggregation function for each performance metric in order to determine the e2e cost of a multihop route.

Additionally, unlike existing anonymity systems that depend on central authorities or directories, our *Application-Aware Anonymity* (A^3) coordinate routing system does not rely on *a priori* trusted nodes or third-party authorities. This lack of centralization enables our system to scale to potentially hundreds of thousands of nodes and offer anonymity that does not depend on the trustworthiness of select nodes or services.

1.3 Research Questions

This dissertation explores the following research questions:

- **How can we scale anonymity networks?** Increasing the size of anonymity networks improves both performance and anonymity. As the ratio of clients to available participants shrinks, so does congestion in the overlay network. Additionally, larger networks likely provide increased geographic and administrative heterogeneity, which in turn impose real-world barriers for colluding eavesdroppers. Our coordinate-based anonymity network is designed to scale to hundreds of thousands of nodes.

- **What are the pertinent e2e performance metrics for anonymous communication?** Previous approaches have attempted to produce anonymous paths with greater e2e bandwidths than that achieved via random participant selection [103, 27]. However, as previously described, there are several applications that impose communication requirements other than bandwidth. We identify several e2e performance metrics in this dissertation, and show how each may be accurately embedded in a coordinate system.

- **How does an initiator create high performance anonymous paths?** We propose *link-based* routing algorithms that utilize coordinate embedding systems to accurately estimate the e2e properties (i.e., latency, jitter, etc.) of potential anonymous paths. Our A^3 architecture empowers initiators to instantiate anonymous paths that meet their specific communication requirements.

- **How do our techniques impact anonymity?** Intuitively, a routing algorithm should select participants uniformly at random from the set of all nodes to maximize anonymity (since knowledge of a portion of the anonymous path leaks no information about the initiator or responder). We purposefully deviate from this approach in order to provide anonymous paths with greater performance. We investigate the tradeoffs between anonymity and performance and analyze the impact on anonymity of our proposed routing algorithms.

- **How can coordinate systems be protected from manipulation?** The systems presented in this dissertation utilize coordinate embedding systems to estimate the cost of potential overlay routes. To produce high performing paths, such estimations must be accurate. The distributed nature of coordinate systems make them particularly vulnerable to insider manipulation [45], as peers can advertise false coordinates or delay responses to measurement probes. This dissertation explores methods of defending coordinate systems from such attacks.

1.4 Contributions

This dissertation makes the following contributions:

- **Anonymous Path Selection Strategies.** *(Chapter 3)* Unlike previously proposed anonymous routing algorithms [27, 103] in which relays are selected based on self-reported *node characteristics* (e.g., available bandwidth), our coordinate-based relay selection strategies estimate the end-to-end performance properties of candidate routes by aggregating *link characteristics* (the properties of connections between nodes). Link characteristics provide better estimates of path properties and support metrics that cannot be represented as a node characteristic (e.g., latency, loss, and jitter). We propose routing algorithms that select anonymous relays based on the estimated cost of paths' constituent links.

- **Coordinate-based Source-Routing for Anonymity.** *(Chapters 4 and 6)* This dissertation proposes the use of coordinate embedding systems for efficiently estimating the e2e cost of potential anonymous routes. We introduce the *Application-Aware Anonymity* (A^3) system [99, 95] that leverages coordinate systems to construct onion paths [82] with tunable anonymity and performance properties. A^3 does not rely on *a priori* trusted nodes, secrets, or services, and is designed to scale to millions of simultaneous users.

- **Veracity.** *(Chapter 7)* This thesis presents *Veracity* [96, 100], a decentralized method of protecting coordinate embedding systems from insider manipulation. Veracity does not depend on any centralized authorities or preshared secrets, and is well-suited for securing the coordinate systems used by A^3.

- **Coordinate-based Source-Routing for Performance.** *(Chapter 8)* We introduce *Contour* [94], an overlay routing system that exploits triangle-inequality violations (TIVs) on the Internet to produce (non-anonymous) paths that perform better than their standard IP routing equivalents.

1.5 Organization

The remainder of this dissertation is presented as follows. In Chapter 2, we describe prior work in anonymous communication. We introduce *link-based* anonymous relay selection algorithms in Chapter 3 and describe practical deployment models in Chapter 4. The anonymity properties of link-based selection strategies are examined in Chapter 5. In Chapter 6, we present the Application-Aware Anonymity (A^3) system that enables initiators to construct high performance anonymous paths. Veracity, a coordinate protection layer that protects the accuracy of A^3's path predictions, is introduced in Chapter 7. In Chapter 8, we propose the Contour detour routing system that leverages virtual coordinate systems to produce high-performance (non-anonymous) paths in overlay networks. We conclude in Chapter 9.

Chapter 2

Related Work

The A^3 anonymity system presented in this dissertation expands upon prior results from the security, anonymity, and networking communities. This Chapter describes the previous research on which our system is built.

2.1 Terminology

Internet *anonymity systems* attempt to hide the identities (network addresses) of one or more communicating parties from eavesdroppers. Typically, messages are passed using application-layer overlay routing with the identities of the initiator and/or the responder obscured using cryptography.

We do not address in this dissertation the interesting, but mostly orthogonal, subject of *covert channels* [51, 70] in which one or more of the communicating parties attempts to conceal the existence of the communication channel itself. Unlike in anonymity networks, any party that is privy to a covert channel can decipher the communicating endpoints. In contrast, anonymity systems hide the identities of the communicating parties even if the eavesdropper intercepts with perfect fidelity. A promising, but unexplored, area of future research is the combination of covert channel techniques with anonymizing networks.

We adopt the following definitions based loosely on Pfitzmann's and Köhntopp's proposal of a unified anonymity terminology [71].

Definition 1. *(Anonymity Set) An* anonymity set *is a finite collection of parties who may be responsible for an action.*

Definition 2. *(Anonymity)* Anonymity *is the state of being not identifiable with absolute (probabilistic) certainty within an anonymity set.*

Definition 3. *(Initiator/Responder Anonymity) A transmission preserves* initiator/responder anonymity *if it does not reveal the identity of an initiator/responder with absolute (probabilistic) certainty. An anonymity system provides* initiator/ responder anonymity *if the actions of its parties do not reveal the identities of the initiators/responders with absolute (probabilistic) certainty.*

Definition 4. *(Perfect Initiator/Responder Anonymity) Let $\mathcal{A}$ be the anonymity set as defined by Definition 1. A transmission α achieves* perfect initiator/responder anonymity *if for all $x \in \mathcal{A}$, the probability that x initiated α is $|\mathcal{A}|^{-1}$. An anonymity system provides* perfect initiator/responder anonymity *if and only if the transmissions of its parties preserve perfect initiator/responder anonymity.*

There is a spectrum in the certainty that an eavesdropper can assign to its belief that a particular party is a communicating party. Michael Reiter and Aviel Rubin introduce the following terminology for assessing the degree of anonymity offered by an anonymity system [83]:

Definition 5. *(Provably Exposed) The eavesdropper can prove that a party participated in the communication.*

Definition 6. *(Exposed) The communicating parties are easily identifiable via packet headers. That is, no anonymity technique is used to hide the initiator or responder. For example, standard IP routing exposes the initiator and responder.*

Definition 7. *(Possible Innocence) There is a nontrivial probability that another party is the communicating party.*

Definition 8. *(Probable Innocence) The probability that a node is a party in the communication is less than* 1/2.

Definition 9. *(Beyond suspicion) The eavesdropper can perceive the communicated messages, but all parties in the system have equal probability of being the initiator/responder. This corresponds to Definition 4.*

Definition 10. *(Absolute privacy) The eavesdropper cannot perceive the transmission of a message. Absolute privacy may be achieved using covert channels [70].*

Ideally, an anonymity system forces an eavesdropper to consider each party in the system as having an equal probability of being the initiator and/or responder for an intercepted message or stream of messages (i.e., "beyond suspicion"). Although some anonymity systems deliver such strong anonymity (a notable example is David Chaum's dining cryptographers scheme [13]), such approaches often assume particular network topologies (e.g., complete graphs) and eavesdropper configurations, and are unsuitable for deployment on the Internet. In practice, the ability of eavesdroppers to perform traffic analysis, timing analysis, and other attacks on Internet anonymity systems (many of which are enumerated in Section 2.4) allows them to construct non-uniform probability distributions over the anonymity set. Hence, A^3's goal is to achieve anonymity that provides the communicating parties with at least "probable innocence".

2.2 Anonymity Systems

In this section, we briefly describe the Crowds [83] and onion routing [82] anonymity systems. A^3 utilizes both techniques to construct anonymous paths. Both Crowds and onion routing are loosely based on Chaum's seminal work on untraceable communication [14].

2.2.1 Crowds

In Crowds, traffic is relayed through a series of participants (proxies) called *jondos*. An initiator initiates an anonymous connection by choosing a small random subset of jondos, F. For each member of the set, the initiator transmits a signed message containing a symmetric key, K_f, that will later be used for encryption and decryption. To protect the confidentiality of the key, the message is encrypted with the public key of the jondo. Formally, the initiator I transmits

$$I \rightarrow \forall f \in F : \{\{K_f\}_{K_{f+}}\}_{K_{I-}}$$

where K_{f+} is the public key belonging to jondo f, K_{I-} is the initiator's private key, $\{X\}_{K_{f+}}$ is the encryption of X using K_{f+}, and $\{Y\}_{K_{I-}}$ denotes that Y has been signed using the private key K_{I-}.

After distributing the symmetric keys to all members of F, the initiator randomly chooses a jondo $j \in F$ and sends the message

$$I \rightarrow j : i, \{R, \texttt{data}\}_{K_j}$$

where i is a key identifier (i.e., it denotes K_j), R is the responder's address, `data` is the contents of the message, and K_j is the symmetric key shared between I and j via the previous protocol step.

With probability $(1 - p_f)$, jondo j will deliver `data` to R. With probability p_f, j will instead forward the message to a *successor jondo*, j', chosen randomly from j's random subset of jondos. In the latter case, j will send

$$j \rightarrow j' : i', \{R, \texttt{data}\}_{K_{j'}}$$

where $K_{j'}$ is the symmetric key shared between j and j' and i' is a key identifier for $K_{j'}$.

Once the route of jondos between I and R has been established, subsequent traffic is sent along that route. At each hop, $\{R, \texttt{data}\}$ is decrypted and re-encrypted using the appropriate shared symmetric keys. Reply traffic traverses the path of jondos in the reverse direction. Routes last until some timeout period, after which the process repeats, forming a new path from I to R.

As described above, the probability that a jondo will deliver a message to the responder is $1 - p_f$. Since the message is delivered as soon as a jondo decides not to forward it to another jondo, the probability of message delivery (i.e., reception by the responder) is geometrically distributed with $p = (1 - p_f)$. Therefore, the probability that the message will reach the responder after traversing i jondos is $p_i = (1 - p)^{i-1}p = (1 - p_f)(p_f)^{i-1}$. Let l be the number of jondos in the path from initiator to responder. The expected value of l is the mean of the geometric distribution, and hence

$$E[l] = 1/p = (1 - p_f)^{-1} \tag{2.1}$$

Note that since $\sum_{i=1}^{\infty} p_i = 1$ (assuming $p_f < 1$), the message will eventually reach the responder.

Assuming the eavesdropper (Eve) intercepts a packet from a party X to the monitored jondo, Eve cannot discern whether X is the initiator (Alice) of the message or a jondo acting on her behalf. Packet capture does not eliminate any nodes from the initiator anonymity set since Alice may be upstream of the point of interception. While the *a posteriori* (after interception) and *a priori* (before interception) initiator anonymity sets are equal, the probability distributions over the sets may differ. If the eavesdropper intercepts packets from X to the monitored jondo, then X is more likely the initiator than all other nodes in the anonymity set. Using Equation 2.1, we can compute the *a posteriori* initiator probability distribution, assuming a uniform *a priori* probability distribution.

Let n represent the number of nodes in Crowds (that is, $n = |\mathcal{A}|$). The probability that X is Alice is $\frac{1}{E[l]} = 1 - p_f$ since this is the eavesdropper's probability of monitoring the first jondo in the path from Alice to Bob. The probability that any other of the remaining $n - 1$ nodes in the anonymity set is Alice is $1 - (1 - p_f) = p_f$.

Since the *a priori* distribution is uniform, each of the remaining $n-1$ nodes is assigned probability $\frac{p_f}{n-1}$ of being the initiator.

Since the hop from the last jondo to the responder is unencrypted, an eavesdropper who monitors a jondo knows with certainty whether the jondo forwards messages to another jondo or to the responder. Otherwise, no information about the responder is learned (i.e., the *a priori* and *a posteriori* responder probability distributions are equal) and all members of the anonymity set are beyond suspicion.

Crowds, like many other Internet anonymity systems, does not provide anonymity against a *global passive adversary* – an eavesdropper who can monitor all traffic on the network. Such a powerful eavesdropper can trace flows forward to identity the responder and use intercept logs to "reverse" Crowd paths to locate initiators.

Crowds is also susceptible to active attacks in which an eavesdropper is also a participant in the system. Crowds relies on hop-to-hop cryptography (the equivalent of link layer encryption applied on top of an overlay network) to protect the identity of the responder. An insider who receives a message of the form $(i, \{R, data\}_{K_j})$ knows K_j and can decrypt the message to learn R. Hence, if the adversary controls any jondo on the path from the initiator to the responder, then the responder is trivially exposed.

2.2.2 Onion Routing and Tor

Onion routing [82] is a source-routed anonymity technique in which the initiator transmits messages via her chosen path of *onion routers* (participants) towards the responder. Messages are multiply encrypted, allowing onion routers to discern only the previous and next hops in the anonymous path. The Tor anonymity network [28, 107] is based on onion routing.[1]

[1] The current Tor implementation does not use strict onion routing. Instead, the initiator uses an iterative "stage" approach in which session keys are distributed to the appropriate onion routing via already established portions of the anonymous path [28].

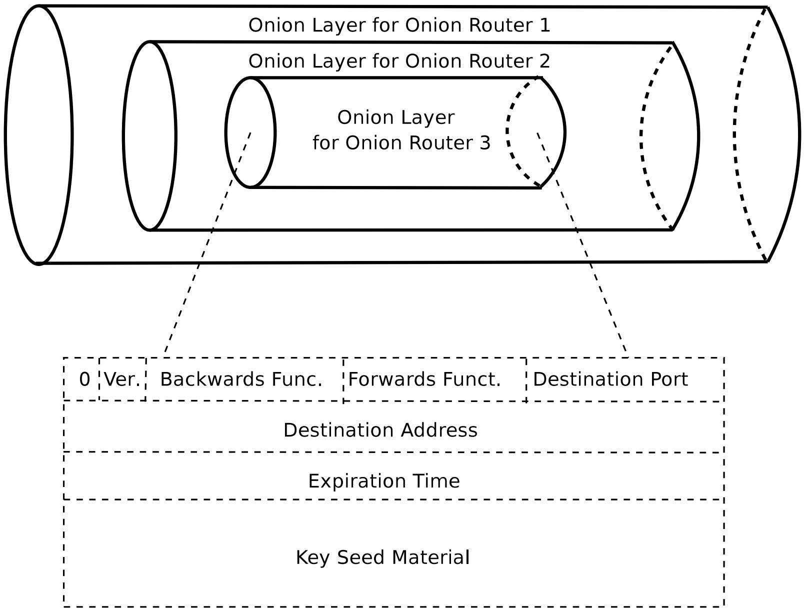


Figure 2.1: A three layered onion. Each onion layer contains the location of the next hop and key material for backwards and forwards encryption, and is encrypted under the public key of the node that receives the onion.

To anonymize her traffic, the initiator (Alice) initiates socket connections through a Tor proxy server, typically running on her machine. The proxy creates a source-routed path consisting of a number of randomly chosen onion routers that terminates at an *onion exit router*. Reply traffic follows the reverse path, traversing through the onion routers and eventually reaching Alice.

There are two phases involved in the establishment of anonymous onion routed sockets: *onion setup* and *data transmission* [82]. In the onion setup phase, the initiator chooses a random sequence of onion routers (i.e., participants) that will form the anonymous path. The initiator's onion router forms a multiply encrypted onion in which each layer of the onion contains key seed material for a particular participant. A three-layered onion is depicted in Figure 2.1. The *key seed material* is hashed to generate symmetric keys for relaying anonymous messages.

Upon receiving an onion, a router peels off and decrypts the outermost layer using its private key. It records its symmetric session key and relays the remaining inner onion layers to the next hop, obtained from the *Destination Host* and *Destination Port* fields. Onions are padded with random data to keep the size of the onions fixed.

In the data transmission phase, Alice repeatedly encrypts her message using the previously distributed symmetric keys, applied in reverse order. For example, if keys α_1, α_2, and α_3 belong respectively to the first, second, and exit onion routers in a path, then message M is encrypted as $\{\{\{M\}_{\alpha_3}\}_{\alpha_2}\}_{\alpha_1}$. Upon receiving an encrypted datagram, an onion router decrypts the outermost layer and relays the result to the next router. Reply traffic is encrypted in the opposite manner: each router encrypts the reply and forwards the result to its predecessor onion router. To mitigate traffic analysis attacks, data is always padded to a fixed length and encrypted containers are used to convey the true length of the data to subsequent onion routers.

Unlike Crowds, onion routing can tolerate the inclusion of adversaries on the anonymous path. Since next hops are encrypted using the public key of each onion router, a malicious participant learns only the identities of the subsequent and the previous hops. Although an adversary on the first hop in an anonymous path trivially knows the identity of the initiator, it cannot infer the identity of the responder. Similarly, an eavesdropper on the last hop knows the responder, but cannot deduce the initiator.

Additional attacks against Tor are described in Section 2.4.

2.3 Path Selection

Routing messages via multihop paths on an application-layer overlay typically yields significantly poorer communication performance as compared to standard IP routing. The e2e performance of an anonymous path, whether measured as latency, bandwidth, or some other metric, is determined by its constituent participants: the latency of an anonymous path is the sum of the latencies of its parts, the bandwidth of a path is determined by the bottleneck imposed by the hop with the least bandwidth, etc. Relaying messages through participants delivers anonymity, but it does so at the expense of performance.

Both anonymity and performance are affected by *path selection* (also called *relay selection*), the algorithm used to select the participants for an anonymous path. If participants are chosen uniformly at random, irrespective of their effects on e2e path performance, anonymity is maximized. In such a case, an adversary has no information on which to bias the probability distribution over the anonymity set. Although choosing participants randomly improves anonymity, such a *topology-agnostic* strategy tends to result in paths with poor performance.

As demonstrated in the following chapter, careful selection of participants may yield high performance paths. However, such discriminatory path selection inherently imposes a non-uniform distribution over the set of candidate relays. Adversaries knowledgeable about the performance of inter-relay links can use the non-uniformity of relay selection to improve its probability of discovering the communicating parties.

An anonymity system's path selection algorithm therefore affects both the anonymity offered by the system as well as the performance of the paths it produces. In systems that allow source-routing, the initiator can potentially choose a point in the anonymity-vs-performance spectrum [103, 99], achieving a desired level of performance while maintaining some degree of anonymity. In contrast, in anonymity systems in which path selection is handled in the network (for example, Crowds), the initiator has no such freedom. For anonymity systems that fall in this latter category, both the performance of the path and the anonymity that it offers is dependent on the selections made by third (and potentially untrustworthy) parties.

As described in later chapters, A^3 uses a source-routing approach, enabling the initiator to tradeoff anonymity and performance [99, 95]. (We defer our discussion of methods for estimating anonymity and performance to later chapters.) Since Tor also uses a source-based approach and attempts to achieve reasonable performance by sacrificing a modicum of anonymity, and because Tor's relay selection algorithm has been shown to enable certain attacks against anonymity [103, 65], we present a simplified version of the Tor path selection strategy below.

Path Selection in Tor Tor's relay selection is a two-step process. First, the initiator builds a list of candidate participants consisting of relays which are marked in the directory server as being stable (having a sufficiently long uptime), valid (running a recent version of Tor), and fast (reporting sufficient available bandwidth). The last participant (exit node) is further constrained to only those relays whose published policies allow them to forward traffic to the responder. To produce high bandwidth routes, Tor's path selection algorithm then sorts relays in increasing order of bandwidth and computes the sum $B = \sum_{i=0}^{|N|-1} b_i$, where b_i is the bandwidth of relay i. The initiator chooses r uniformly at random from $[0, B)$ and selects the node with index k as a relay, where k is the largest integer such that $\sum_{i=0}^{k-1} b_i \leq r$. The initiator repeats this procedure to select each relay in the anonymous circuit [27].[2]

2.4 Attacks on Anonymity

There have been several well-publicized attacks against Internet anonymity systems. Since A^3 borrows concepts from onion-routing and Tor, the attacks described in this section focus on such anonymity designs.

Although the taxonomy presented below is intended to highlight several of the more severe attacks against anonymity, it should not be considered a complete enumeration of all possible attacks.

[2] In practice, Tor may apply different weights for entry and exit nodes. For simplicity, we assume that all nodes may function as entry or exit relays.

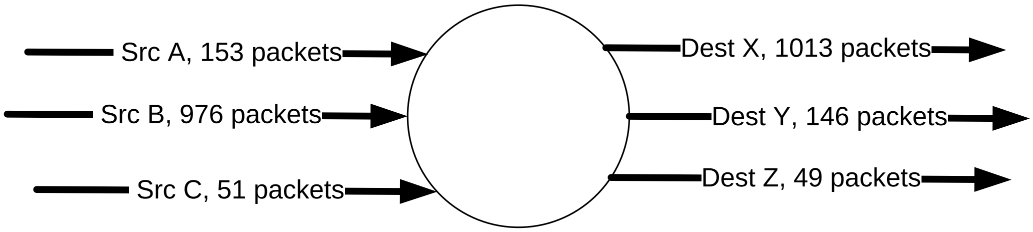


Figure 2.2: Example counting attack.

2.4.1 Counting Attacks

In anonymity systems that utilize mixing [14], participants attempt to aggravate traffic analysis by enqueuing inbound messages and relaying traffic in randomly ordered bursts. Example mix networks include ISDN-Mixes [72], Web Mixes [9], JAP [35], and onion-routing [82]. The delay between message reception and forwarding makes it more difficult for an eavesdropper to associate connection streams. Figure 2.2 shows an example mixing in which a participant receives packets from three sources (A, B, and C), buffers the messages, and outputs packets to three destinations (X, Y, and Z).

Serjantov and Sewell describe a passive *packet counting attack* in which an eavesdropper counts the number of incoming and outgoing packets traversing a monitored participant for a set time period [92]. If the number of incoming packets for one connection (identified by IP address and port number) approximately equals the number of outgoing packets for another connection, then the attacker can assume with high confidence that the two connections are part of the same anonymous path. For example, an adversary can easily infer the mapping $\{A \rightarrow Y, B \rightarrow X, C \rightarrow Z\}$ from the counts depicted in Figure 2.2.

Such attacks work best for *lone connections* in which only a single connection traverses a monitored link. Packet counts cannot be easily correlated if multiple connections are multiplexed on a link, since messages belonging to each connection may be distributed amongst one or more links exiting the mix. Since larger networks provide more opportunity for lone connections, larger networks are *more vulnerable* to attack than smaller anonymity networks in which traffic from multiple anonymous connections are regularly multiplexed over shared links.

2.4.2 Low-Latency Attacks

Mix-based anonymity networks incur heavy performance penalties by storing messages for relatively long periods before forwarding them to their next hops. Tor and other low-latency anonymity networks opt to forgo mixing in favor of reducing the e2e latencies of anonymous paths.

To reduce queuing delay while providing a measure of fair service, Tor enqueues incoming messages according to streams (i.e., anonymous paths). Onion routers maintain a bin for each stream, assigning arriving traffic to the appropriate bin. Bins are serviced in a round-robin fashion, with messages being dequeued and relayed to their subsequent hops [28].

Although the round-robin approach provides fair queuing, it enables the following subtle attack on anonymity [64]: Since Tor maintains a bin for each anonymous stream and services them in a round-robin fashion, the amount of time between bin accesses (i.e., latency) is thus a function of the number of bins. Therefore, the addition of an anonymous stream at a given onion router impacts the latency of all other streams traversing that router. An adversary can exploit this behavior to test whether a particular router is upstream of a monitored anonymized link. To perform the attack, the adversary constructs her own anonymous path through the candidate router. If traffic sent on the adversary's anonymous path is correlated to an increase in latency of the monitored connection, then that connection likely flows through the candidate router. Recent work by Evans *et al.* improves upon this attack by creating long cyclical paths through the targeted relay [33].

2.4.3 Predecessor Attacks

First described by Reiter and Rubin [83] and subsequently analyzed in more depth by Wright *et al.* [116], the *predecessor attack* enables an adversary to discern the identity of an initiator. The attack exploits the observations that anonymous conversations are often longstanding and their underlying connections are subject to frequent *resets*. Here, we make the distinction between a longstanding *conversation* between the initiator and the responder and the more transient anonymous *connections* over which the conversation is carried. Resets, or interruptions in connections due to churn or other network effects, force the initiator to establish a new connection in order to continue his conversation. Note that the initiator and responder remain fixed between resets whereas a new set of participants are chosen for each connection.

If an adversary controls some fraction of the nodes in the anonymity network, she can count the number of times each predecessor node participates in an anonymous connection for a given conversation. (The predecessor attack is contingent on the adversary's ability to discern session-identifying information in transported messages.) Maintaining such tallies reveals the initiator's identity since the initiator is more often the predecessor than any other node.

The countermeasure to the predecessor attack is the use of *entry guards*, fixed participants that serve as the first hop in all anonymous paths [116]. Each initiator chooses an entry guard through which it routes all paths. Since the entry guard is consistently the first hop of all anonymous paths, nodes under the adversary's control do not directly communicate with the initiator, and at worst identify only the entry guard, effectively thwarting the attack.

2.4.4 Timing Attacks

Tor provides a mechanism for responder anonymity called *hidden services* in which the initiator and responder communicate via a rendezvous point in the network [28].

Although the responder participates in the anonymous connection, his identity is obscured by the rendezvous point. Hidden services are useful, for example, to host information (e.g., web pages) without being identifiable.

Steven Murdoch identified a mechanism for locating Tor Hidden Services using clock skew measurements [63]. To conduct the attack, an adversary induces load on the hidden service by making frequent requests. The high load causes temperature changes on the server, which in turn results in increased clock skew. An attacker can observe the changes in clock skew of a set of potential hosts to deduce the identity of the responder. The effectiveness of this approach was recently improved by Sebastian Zander and Steven Murdoch using synchronized sampling techniques [119].

Although Murdoch's timing analysis attack is targeted at breaking the anonymity of hidden services, similar timing analyses can be applied to identify the initiator and the responder of an anonymous circuit. If an eavesdropper controls both the first and last participants in an anonymous path, she can correlate packets traversing her two hosts to determine that they belong to the same anonymous path [63, 103]. (Recall that a design goal of Tor is to limit each participant to knowing only the previous and next hops.) Since the first participant knows the identity of the initiator and the last participant has knowledge of the responder, an eavesdropper who controls both the first and last participants knows the identities of the communicating parties. Although one or more participants not under the adversary's control may be located between the first and last hops on the anonymous path, these intermediary routers do not perturb timing information sufficiently to mitigate the attack. Of course, such "middle" participants may purposefully apply jitter to the channel to aggravate timing analysis, but by doing so, they increase the latency of the path. To provide low-latency anonymity, Tor does not take such measures.

2.4.5 Path Selection Attacks

Previously proposed relay selection techniques have focused on improving the bandwidth of generated paths [27, 103]. To produce high bandwidth routes, the Tor [28] path selection algorithm sorts relays in increasing order of bandwidth and computes the sum $B = \sum_{i=0}^{|N|-1} b_i$, where b_i is the bandwidth of node i. The initiator chooses r uniformly at random from $[0, B)$ and selects the node with index k as a relay, where k is the largest integer such that $\sum_{i=0}^{k-1} b_i \leq r$. The initiator repeats this procedure to select each relay in the anonymous circuit [27].[3]

Øverlier and Syverson first identified that Tor's path selection algorithm is susceptible to manipulation [68]. By falsely advertising high bandwidths, nodes under an adversary's control can exploit the weighted probability distribution and increase their chances of being selected. If multiple nodes under the attacker's control are selected as relays, the adversary can apply a circuit-linking algorithm [8] or perform timing analysis [63] to discern whether two of its relays reside on the same path. (Tor is designed to restrict each relay to knowing only the previous and next hop [28].) If the attacker controls the first and last relays in an anonymous path, he defeats anonymity since the first and last relays respectively know the identities of the initiator and responder. Bauer *et al.* demonstrate that when an adversary controlled just six of 66 nodes in a Tor deployment on PlanetLab [74], the attacker compromised more than 46% of all anonymous paths [8].

Snader and Borisov [103] propose two modifications to Tor to defend against Øverlier *et al.*'s attack. First, to prevent nodes from reporting false bandwidths, relays report the observed bandwidths of peer relays to the directory server. When queried for a node's bandwidth, the directory server reports the median of the node's observed measurements. Second, Snader and Borisov introduce a more tunable

[3] In practice, Tor may apply different weights for entry and exit nodes. For simplicity, we assume that all nodes may function as entry or exit relays.

weighting system in which the initiator can tradeoff between anonymity and performance. They define the family of functions

$$f_s(x) = \begin{cases} \frac{1-2^{sx}}{1-2^s} & \text{if } s \neq 0 \\ x & \text{if } s = 0 \end{cases} \tag{2.2}$$

where s is a parameter chosen by the initiator that allows it to tradeoff between anonymity and performance. After having ranked the relays by bandwidth, the initiator chooses the relay with index $\lfloor n \cdot f_s(x) \rfloor$, where n is chosen uniformly at random from $[0, 1)$. By applying higher values of s, the initiator is able to more heavily bias her selections towards bandwidth. If $s = 0$, a relay is chosen uniformly at random [103]. Each relay is selected independently and without replacement according to the distribution imposed by Eq. 3.1.

Snader and Borisov's defense relies on *opportunistic measurements* – relays report the observed bandwidths of their peers [103]. There are unfortunately disadvantages of such an approach. First, a relay can report opportunistic measurements only when it participates in an anonymous circuit with a peer. Transmitting the observation to a directory server effectively informs the server of the existence of the circuit as well as the identities of the two relays that constitute one of its hops. Given that directory servers may be malicious, revealing segments of the path is undesirable. Second, the directory cannot discern whether reported measurements are truthful. Colluding malicious relays may (falsely) report that members of their coalition have high bandwidth. If there are a sufficient number of attackers to influence the median of a relay's measurements, then Øverlier *et al.*'s attack becomes feasible. Finally, as noted in Murdoch's recent work [65], attackers may have access to large botnets and may therefore join the anonymity network with relays that have sufficient bandwidth to attract peers. The use of opportunistic measurements attempts to protect against false self-reported measurements, but does not prevent an attacker from acquiring high performing nodes to attract traffic. As we show in Chapter 3, link-based measurements inherently reduce the attacker's ability to influence path selection.

In the following chapters, we present novel link-based relay selection strategies to achieve high performance anonymous paths. Our link-based algorithms drive the Application-Aware Anonymity (A^3) platform that we introduce in Chapter 6, as well as the Contour detour routing system presented in Chapter 8.

Chapter 3

Link-Based Path Selection

Existing approaches [28, 27, 103] to producing high performance anonymous paths have focused exclusively on *node characteristics* – performance metrics (i.e., bandwidth) that may be attributed to individual relays. Node-based relay selection strategies randomly select relays according to a nonuniform probability distribution biased by the relays' node characteristics.

In link-based path selection, the e2e performance of a path is computed by aggregating the cost of all links that comprise the path, where cost is defined in terms of *link characteristics* such as latency, loss, and jitter. (While bandwidth is a node-based characteristic, it can also be represented as a link characteristic by considering the measured available bandwidth on a link connecting two nodes.) The use of link rather than node characteristics enables not only more flexible routing, as initiators can construct anonymous routes that meet more specific communication requirements, but (as we show in Chapter 5) also better protects the identities of the communicating parties.

3.1 Weighted Path Selection

Our link-based path selection algorithm, Weighted, operates in two phases.

Metric	**Path cost**
Latency / RTT	$\sum_{i=1}^{h} d_i$
Bandwidth	$\min(d_1, d_2, ..., d_h)$
Loss rate	$1 - \prod_{i=1}^{h}(1 - d_i)$
Jitter (variance)	$\sum_{i=1}^{h} d_i$
Autonomous System (AS) Traversals	$\sum_{i=1}^{h} d_i$

Table 3.1: Link concatenation operators. The distance between successive links is denoted as $d_1, d_2, ..., d_n$. We define jitter as the variance (σ^2) of a random variable describing packet inter-arrival time. Assuming that the jitter of two successive links are independent, then the aggregate variance is their sum, since $\sigma(X+Y)^2 = \sigma(X)^2 + \sigma(Y)^2$ for independent random variables X and Y.

In the first phase, the initiator rapidly generates (but does not instantiate) candidate paths consisting of three relays chosen uniformly at random without replacement. The initiator computes the e2e cost of each generated candidate path using a *link concatenation operator* (see Table 3.1). For example, the e2e bandwidth of a path is the minimum of the bandwidths of its links, whereas the latency of the route may be estimated by summing the latencies of its hops. Our approach may be extended to define the performance of a path in terms of *multiple metrics* by assigning weights to each metric in a manner that reflects its importance as determined by the initiator. The e2e path cost estimate is then calculated as the weighted average over the cost estimates for each individual metric.

In this chapter, we assume that initiators have knowledge of the network distances between any two relays and can therefore calculate path costs. In the following chapter, we relax this assumption and describe efficient methods for disseminating pairwise distance information. For the remainder of this chapter, however, we make the simplifying assumption that all nodes know all pairwise distances.

In the second phase, the initiator sorts the candidate paths by their cost estimates. Using the family of functions introduced by Snader and Borisov [103]:

$$f_s(x) = \begin{cases} \frac{1-2^{sx}}{1-2^{s}} & \text{if } s \neq 0 \\ x & \text{if } s = 0 \end{cases}$$

the initiator instantiates the candidate path with index $\lfloor n \cdot f_s(x) \rfloor$, where n is chosen uniformly at random from $[0, 1)$. As with Snader's and Borisov's algorithm, a larger value of s more heavily weighs path selection in favor of performance. When $s = 0$, each randomly generated path is equally likely to be chosen. For clarity, we will refer to the special case in which $s = 0$ as using the Uniform selection strategy.

3.2 Assumptions and Limitations

The Weighted relay selection strategy estimates the e2e performance of potential anonymous routes by aggregating the costs of their constituent hops. Since the anonymity service's queuing and servicing of messages affect communication performance, measurements should be carried out at the application-layer. That is, the cost of routing between relays R_1 and R_2 should encompass not only network effects, but also the message processing expense incurred by R_1 to transmit a message and R_2 to process it.

To be effective, link-based routing requires that path performance (whether it be measured by bandwidth, latency, jitter, etc.) be due primarily to network effects. If, however, local effects at end nodes (e.g., congestion or queuing delay) dominate performance, then link-based path selection is less effective since the savings gained from optimizing link costs is overshadowed by node effects.

The performance and anonymity results in the following chapters assume path performance is dictated by the network rather than end-host effects. Such an assumption is unlikely to hold for more centralized networks (specifically, Tor [26]) in which the number of clients may exceed the number of relays by several orders of magnitude. With such a top-heavy ratio of clients to relays, congestion at the relays becomes the dominant factor in performance.

Link-based routing is better suited for anonymity networks in which network characteristics play an important role in determining the performance of anonymous paths. Specifically, the Application-Aware Anonymity (A^3) framework introduced in Chapter 6 incorporates a peer-to-peer (p2p) architecture to minimize congestion. A p2p design has two important advantages over more centralized approaches. First, since each node is both a potential initiator and relay, a p2p anonymity system inherently balances the number of relays and clients. Provided that the relay selection strategy does not significantly favor certain subsets of nodes (a scenario we investigate later in this chapter), a more distributed architecture alleviates the problem of congestion, making network performance the determining factor in anonymous path performance. Second, p2p systems are better suited for the traffic that traverses anonymity systems. In particular, McCoy *et al.* have shown that BitTorrent [15], a p2p file sharing service, accounts for more than 40% of Tor's bandwidth [60]. Using a p2p anonymity network to serve p2p traffic is a natural fit.

Although Weighted does not require a p2p architecture, its usefulness is dependent on whether link-based cost estimations accurately reflect the e2e performance of anonymous paths. It should be emphasized, however, that in the extreme case in which performance is determined solely by node characteristics, link-based selection becomes equivalent to (and no worse than) node-based techniques. That is, congestion at relay R may cause the cost of all links $X \rightarrow R$, for all other relays X ($X \neq R$), to be equal to one another and a function only of R's congestion. In such a case, weighting by link costs clearly becomes equivalent to weighting based on node costs.

In the remainder of this thesis, we assume that the effects of congestion and queuing delays at individual hosts do not overshadow the cost of routing between nodes. The scalable A^3 p2p architecture introduced in Chapter 6 supports such an assumption.

3.3 The Case for Link-based Selection

In this section, we present the case for link-based path selection. We demonstrate that link-based anonymous routing is flexible, enabling high performance paths, whether performance be quantified in terms of bandwidth, latency, or jitter.

We first consider an *oracular* model in which all measurements (node or link) in the network are known to the initiator. This enables us to compare node- and link-based path selection strategies irrespectively of their measurement techniques. We explore actual implementation strategies in Chapter 4.

Performance Analysis Our performance analysis highlights two main benefits of link-based path selection over existing node-based techniques. First, link-based techniques support a variety of performance metrics, hence offering greater flexibility. In particular, the WEIGHTED selection strategy produces paths with low latency and jitter, few autonomous system (AS) traversals, and high bandwidth. Second, as with recently proposed node-based approaches [103], our link-based relay strategy enables the initiator to carefully tradeoff between anonymity and performance.

Our performance analysis is carried out using a trace-driven path simulator that takes as input an $N \times N$ matrix describing the pairwise network distances (i.e., latency, bandwidth, etc.) between relays. The pairwise link distances used as input to the simulator are obtained from actual network traces [47, 117] as well as our own measurements carried out on the PlanetLab testbed [74]. Since the performance and security of link-based path selection is influenced by the underlying topology, we analyze the results of generating 150 anonymous paths between each of the $N(N-1)$ pairs of relays. That is, for each pair of relays, we generate anonymous paths between the pair using the remaining $N-2$ nodes in the dataset as potential relays. To produce each path, WEIGHTED generates (but does not instantiate) 150 candidate paths[1] before randomly selecting the chosen path according to the weighted (e.g., by

[1] In practice, the number of candidate paths can far exceed 150, as commodity processors can

Dataset	Metric	Nodes	Description
King [47]	Latency	500	Pairwise latencies captured using the King method [42]
Meridian [114]	Latency	500	Pairwise latencies obtained using Meridian [114]
S^3-BW [117]	Available Bandwidth	365	Pairwise bandwidths from PlanetLab measured using PathChirp [86]
PL-ASes	AS Traversals	156	Pairwise number of AS crossings on PlanetLab measured using traceroute
PL-Jitter	Jitter (variance)	259	Pairwise jitter (variance of interarrival times of 30 pings) on PlanetLab
Tor-BW	Available Bandwidth	500	Available (also called “observed”) bandwidth of 500 Tor nodes, obtained from Tor directory servers

Table 3.2: Network datasets used to evaluate link-based relay selection.

bandwidth) probability distribution.

Table 3.2 describes the trace-driven datasets used as input to our simulator. The King [47], Meridian [114], and S^3-BW [117] datasets are based on measurements obtained from prior publications and are commonly used in the networking research community; PL-ASes and PL-Jitter represent newer metrics that are novel to this work. Due to the lack of existing published traces on these metrics, we conducted our own measurements using geographically distributed PlanetLab nodes. The Tor-BW dataset was culled from Tor's directory servers.

Since simulation time grows geometrically with network size, only the pairwise measurements for the first 500 relays from the King, Meridian, and Tor-BW datasets are used as input to the simulator. The remaining datasets contained fewer than 500 nodes, and are used in their entirety.

3.3.1 Bandwidth

Figure 3.1 shows the bandwidth improvement resulting from using Weighted on the S^3-BW dataset. When $s = 9$, Weighted more than doubles the median available bandwidth over all pairwise paths to 42.3 Mbps, compared to 20.1 Mbps when relays are selected uniformly at random (Uniform). (Recall that relay selection is weighted

generate thousands of candidate paths per second. We restricted Weighted to consider only 150 candidate paths since Weighted is invoked at least $150 \cdot N(N-1)$ times per simulation.

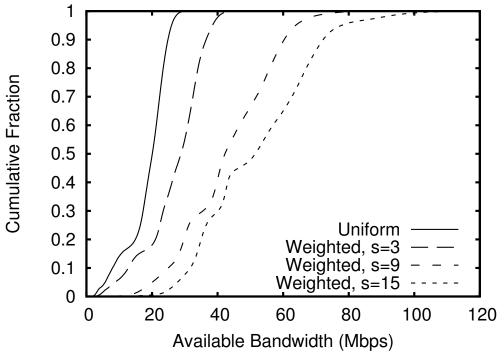


Figure 3.1: E2e available bandwidth using the S^3-BW dataset.

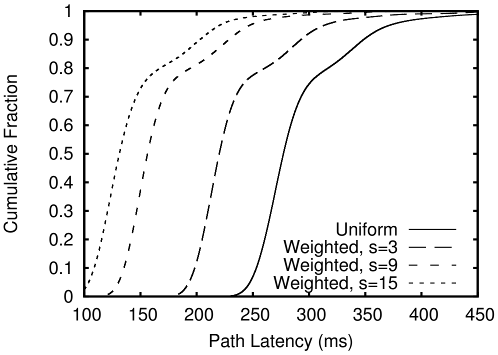


Figure 3.2: E2e path latencies using the King dataset.

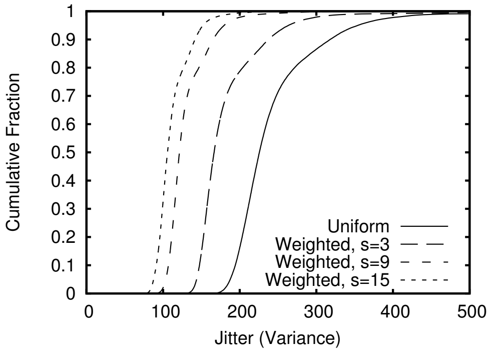


Figure 3.3: E2e jitter using the PL-Jitter dataset.

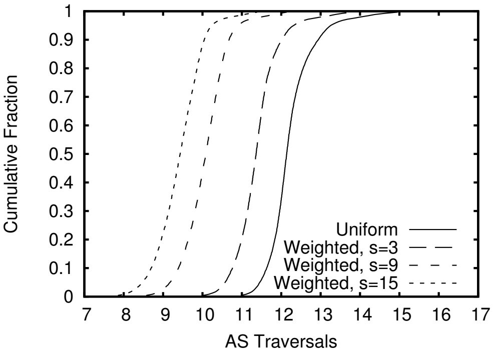


Figure 3.4: E2e AS traversals using the PL-ASes dataset.

more heavily towards performance when s is increased.) The ability to provide high performance bandwidth paths using link-based relay selection is particularly interesting, given that bandwidth is often perceived as a node characteristic [2, 50]. Bandwidth may, of course, be represented as a link characteristic (as is the case in the S^3-BW dataset). This latter characterization enables more flexible routing, as bandwidth bottlenecks may result from Internet routing policies rather than node capacities.

3.3.2 Non-bandwidth Metrics

Figures 3.2 through 3.4 demonstrate WEIGHTED's ability to produce high performance paths for non-bandwidth metrics. The median e2e latency of the anonymous paths formed using `Uniform` is 277.2ms (Figure 3.2). The median latency decreases by 20.6% to 220.1ms when $s = 3$ and by 52.7% to 131.2ms when $s = 15$. Additionally, WEIGHTED decreases the percentage of high latency paths: 93.0% of paths produced via UNIFORM have latencies of 250ms or greater compared to just 22.5% of routes generated using WEIGHTED with $s = 3$.

Jitter, defined as the variance in interarrival times (measured in ms) of 30 ping messages, significantly decreased using WEIGHTED. As shown in Figure 3.3, the median jitter decreased by 26.2% when $s = 3$ and by 46.3% when $s = 9$.

It may also be advantageous to minimize the number of AS crossings in an anonymous path, both to decrease the probability that a given AS can observe multiple hops in the path [34] and also to potentially achieve greater path performance (since routing within an AS is typically low-latency and high-bandwidth). Although analyzing the relationships between AS traversals, anonymity, and performance is beyond the scope of this dissertation, we include the metric here to emphasize the flexibility of link-based routing. Figure 3.4 shows the cumulative distribution of AS traversals for anonymous paths. Using `Uniform`, 66% of anonymous paths traversed 12 or more ASes. In contrast, when $s = 3$ and $s = 9$, only 10% and 0.3% of their respective paths crossed 12 or more ASes.

Chapter 4

Practical Link-Based Path Selection

Our analysis in the previous chapter assumed that the initiator has knowledge of pairwise distances between potential relays. In practice, maintaining pairwise distances will require $O(N^2)$ in communication and network state, hence imposing a significant overhead on the anonymity network.

One practical solution to the above challenge is via the use of *network coordinate systems* that enable the pairwise distances between all participating nodes to be estimated to high accuracy with low overhead. Network coordinate systems, such as Vivaldi [19], PIC [16], NPS [66], and Big Bang Simulation [93] map each relay to n-dimensional coordinates such that the Euclidean distance between two relays' coordinates corresponds to the actual network distance between the pair. Although their individual implementations differ, coordinate systems use distributed algorithms in which each participant periodically measures the distance between itself and a randomly selected peer. By comparing the empirical measurement with the coordinate-based Euclidean distance estimation, a relay can adjust its coordinate either towards (in the case of over-estimation) or away from (for under-estimation) its peer's coordinate.

Network coordinate systems are well-suited for link-based relay selection, effectively linearizing the quantity of information that must be stored and communicated: knowledge of each relay's coordinate is sufficient for estimating the pairwise distances between them. Virtual coordinate systems are lightweight, requiring little bandwidth overhead, and adapt quickly to changes in the network [19]. Additionally, these systems have proved to operate efficiently at Internet scale. For example, the Vuze [109] BitTorrent [15] client currently operates a coordinate system consisting of more than one million nodes [52]. Finally, the *Veracity* protection system, introduced in Chapter 7, prevents misbehaving relays from falsifying their coordinates to attract traffic, ensuring the accuracy of advertised coordinates.

4.1 Background: Vivaldi Coordinate System

Since the Vivaldi coordinate system has been the subject of significant study [96, 100, 45, 44] and is the underlying coordinate system used by our A^3 implementation, we briefly describe its operation in this section. Our link-based relay selection strategies are compatible with any embedding system in which pairwise distances can be estimated using peers' coordinates. (Other virtual coordinate systems [93, 16] function similarly to Vivaldi.)

Vivaldi uses a fully distributed spring relaxation algorithm, requiring no fixed network infrastructure and no distinguished nodes. The system envisions a spring between each pair of nodes, with the resting position of the spring equaling the network latency between the pair. At any point in time, the distance between the nodes in the coordinate space determines the current length of the spring connecting the nodes.

Nodes adjust their coordinates after collecting published coordinate and latency measurements from a randomly chosen neighbor. Consider a node i that wishes to update its coordinate C_i. It picks a randomly chosen neighbor j, retrieves its coordinate C_j and performs a round-trip measurement RTT_{ij} from itself to j. The squared error function,

$$E_{ij} = (RTT_{ij} - ||C_i - C_j||)^2 \tag{4.1}$$

(where $||C_i - C_j||$ is the distance between their coordinates) denotes the *estimation error* between the coordinates of i and j. Using Vivaldi's spring relaxation algorithm, E_{ij} reflects the potential energy of the spring connecting the two nodes. Vivaldi attempts to minimize the potential energies over all springs. In each timestep of the algorithm, nodes allow themselves to be pulled or pushed by a connected spring. The system converges when the squared error function (i.e., the potential energies) is minimized below a threshold.

Our coordinate-based anonymity system is novel in its use of coordinate systems to embed non-latency metrics. We generalize the squared error function (Equation 4.1) as

$$E_{ij} = (D_{ij} - ||C_i - C_j||)^2 \tag{4.2}$$

where D_{ij} denotes an arbitrary network measurement (e.g., latency, jitter, loss) between nodes i and j. Since *closeness* between two coordinates denotes lower cost (for example, small link latency), link bandwidths (in which greater distances are more desirable) must be encoded. If b_{ij} denotes the bandwidth between nodes i and j, then $D_{ij} = B - b_{ij}$, where B is the maximum possible bandwidth in the network.

4.2 Metrics

To assess the accuracy of a virtual coordinate, we measure the **median error ratio** of a node n_i, defined as the median over the *error ratios*

$$\frac{\left|D(n_i, n_j) - ||C_{n_i} - C_{n_j}||\right|}{D(n_i, n_j)} \tag{4.3}$$

between n_i and all other nodes n_j ($n_i \neq n_j$). Coordinate estimation errors are due to the presence of network triangle inequality violations that cannot be expressed using Euclidean geometry.

Conceptually, Equation 4.3 computes the difference between the computed distance between n_i and n_j based on coordinates ($||C_{n_i} - C_{n_j}||$) and the actual measured distance (denoted by $D(n_i, n_j)$). The accuracy of a node's coordinate increases inversely with its median error ratio. The use of median provides an intuitive measure of a coordinate's accuracy and is more robust than average to the effects of outlier errors. Previous approaches [19, 66, 118] define similar metrics.

To gauge the accuracy of the system as a whole, we define the **system error ratio** as the median over all peers' median error ratios. To show lower performance bounds, we also consider the **90th percentile error ratio** – i.e., the 90th percentile of nodes' median error ratios.

4.3 Performance Impact of Coordinate Systems

Since network distances cannot be perfectly embedded using Euclidean geometry, estimation errors may impact the performance of anonymous paths. To quantify the accuracy of coordinate systems, we executed the Vivaldi [19] embedding system on the `King`, `PL-ASes`, and `PL-Jitter` datasets. We utilized the Vivaldi implementation from the Bamboo DHT [7], configured to use a five dimensional coordinate plane.[1]

Figure 4.1 plots the cumulative distribution function (CDF) of nodes' median error ratios after stabilization for the three datasets. As expected, the latency dataset results in lowest error ratios. The system error ratio for the `King` dataset is just 10.9% (6.1ms).

Although intended to embed latencies, our results show that Vivaldi is also effective at embedding non-latency metrics. The system error ratios for the `PL-ASes` (AS counts) and `PL-Jitter` (jitter) datasets are 21.5% (0.45 ASes) and 29.6% ($\sigma^2 = 11.5$), respectively. The higher estimation errors (as compared to `King`) are due

[1] Although comments in the Bamboo source code recommend the use of five dimensional coordinates [7], similar results were achieved using three dimensional coordinates.

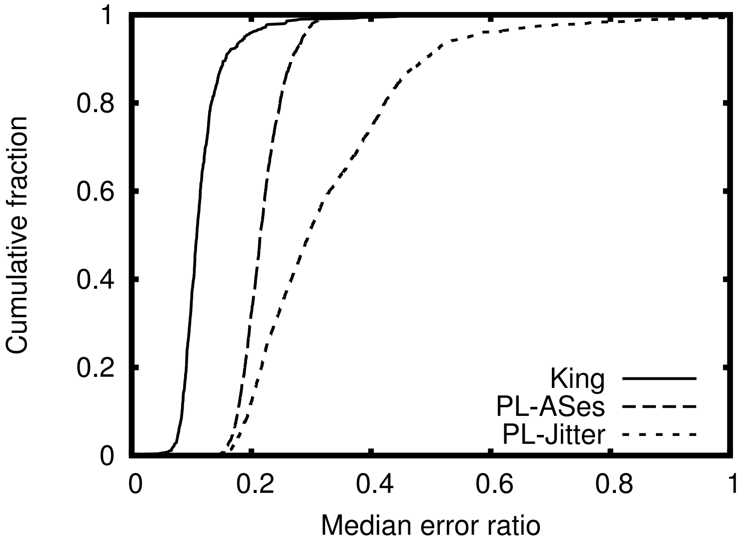


Figure 4.1: CDF of median error ratios for the `King`, `PL-ASes`, and `PL-Jitter` datasets.

to the higher fraction of triangle inequality violations in the `PL-ASes` and `PL-Jitter` datasets. However, as we show below, the achieved error ratios are sufficiently low to generate high performance anonymous paths.

Using the `King` dataset, Figure 4.2 shows the impact that latency estimation errors have on the e2e performance of anonymous paths. The figure compares the e2e performance of paths produced by WEIGHTED when actual distances ("Actual") and coordinate-based estimations ("Estimated") are used. As is apparent from the figure, the use of virtual coordinates to estimate distances does not significantly degrade the performance of anonymous paths. For example, when $s = 15$, the median e2e path latency is 131.2ms using actual network distances; the use of virtual coordinates incurs a modest 8% increase in latency, resulting in paths with a median e2e latency of 141.9ms (still far below the 277.1ms median obtained by UNIFORM).

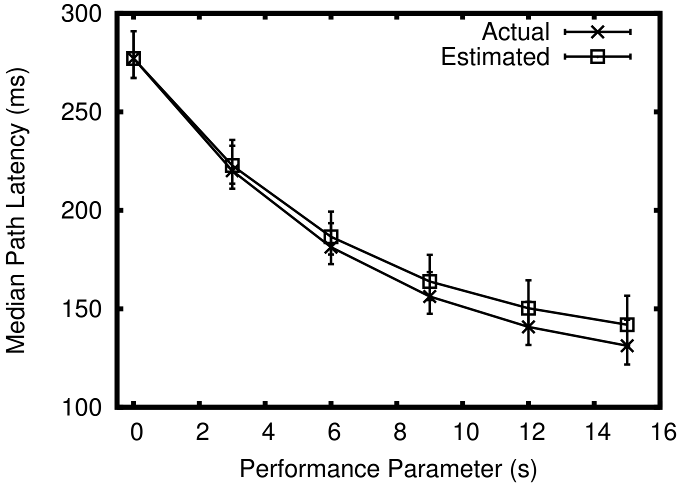


Figure 4.2: Path performance for the King dataset using actual network distances ("Actual") and coordinate-based distance estimations ("Estimated"). Points denote median values with errorbars representing the standard deviation.

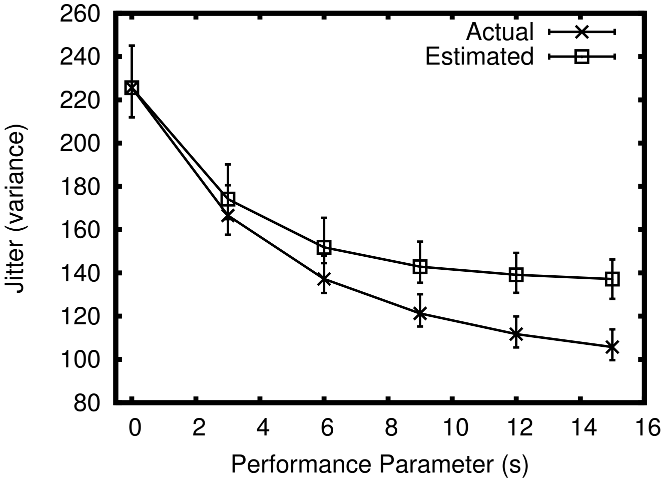


Figure 4.3: Path performance for the PL-Jitter dataset using actual network distances ("Actual") and coordinate-based distance estimations ("Estimated"). Points denote median values with errorbars representing the standard deviation.

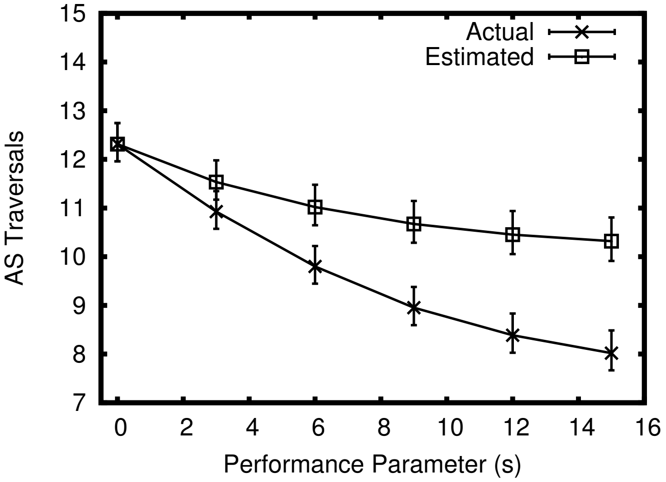


Figure 4.4: Path performance for the PL-ASes dataset using actual network distances ("Actual") and coordinate-based distance estimations ("Estimated"). Points denote median values with errorbars representing the standard deviation.

Although jitter and AS traversals do not embed as seamlessly as latency, the use of coordinate systems to estimate these non-latency metrics achieves paths with significantly less jitter and AS traversals than Uniform selection (Figures 4.3 and 4.4). For example, when $s = 15$, Weighted decreases jitter by 39.2% and the number of AS traversals by 16.2% compared to Uniform.

Bandwidth estimation Unmodified coordinate systems have been shown to be ineffective at estimating pairwise bandwidths due to the high incidence of TIVs in bandwidth measurements [111, 79, 80]. Fortunately, there have been a number of recent promising proposals that enable more accurate bandwidth estimations. In particular, there have been several attempts to identify links that cause severe network TIVs [111, 56, 53], enabling initiators to avoid them when forming paths. Recent work [79, 80] has directly addressed the problem of bandwidth embeddings, introducing techniques for embedding bandwidth distances in tree structures. Their results show that pairwise PlanetLab bandwidths can be embedded with a median error ratio of approximately 0.25 [79]. Although these coordinate embedding system refinements have not been integrated into our implementation, we note that WEIGHTED is agnostic to the underlying coordinate mechanism. That is, a more accurate coordinate system leads to more accurate e2e path estimations, which (as indicated by Figures 4.2 through 4.4) produce higher performance paths. Hence, any improvements in coordinate system design consequently strengthens the performance of coordinate-based relay selection.

4.4 Coordinate System Security

The distributed nature of coordinate systems make them vulnerable to manipulation if not properly defended. Malicious relays may advertise false coordinates or delay measurement probes, either to make themselves appear more favorable or to cause disorder in the system. For example, when 30% of Vivaldi nodes lie about their coordinates, its system error ratio increases by a factor of five [45]. When attackers collude, a coalition that controls as little as 20% of the network increases estimation error by 718% [96].

Fortunately, practical techniques exist that mitigate such attacks [16, 89, 44, 118, 100, 96]. In Chapter 7, we introduce *Veracity* [100, 96], a fully distributed coordinate system protection layer that prevents colluding malicious nodes from advertising false coordinates. Veracity is particularly well-suited for anonymity systems due to its decentralized architecture and its lack of reliance on *a priori* shared secrets or trusted nodes.

4.5 Locating the Responder

To estimate e2e path performance, the initiator must predict the distance between the exit relay (the last participant in the anonymous path) and the responder. If the responder is an active participant in the anonymity system, he can also participate in the embedding system and maintain a coordinate. In such cases, the initiator can estimate the cost of the last hop as the Euclidean distance between the two pertinent coordinates. However, if the responder is not an active participant (for example, the responder may be a website) and therefore has no coordinate, then the initiator requires some mechanism of estimating the final distance.

In this section, we propose a number of solutions the initiator may employ to estimate the distance between the exit relay and the receiver.

4.5.1 Closest Relay Services

The initiator may locate the closest relay to the responder using publicly available network information services. For example, OASIS [38], ClosestNode [115], and iPlane [57] all provide interfaces for resolving the closest server to any given IP address. The initiator can anonymously query such services to locate the relay that is nearest to the responder. (Such a procedure is analogous to anonymously querying DNS servers to resolve hostnames to IP addresses.) The closest relay can then proxy requests between the exit relay and the responder.

4.5.2 AS Path

Alternatively, the initiator could derive AS paths between potential exit relays and the responder using data from the University of Oregon's Route Views project [1]. Here, the initiator uses the heuristic that shorter AS paths likely yield better performance than longer paths. Such an approach has the advantage that it can be executed offline by periodically downloading the Route Views dataset. Since AS path estimation is an expensive operation, it is best suited for applications in which path generation is infrequent and longstanding.

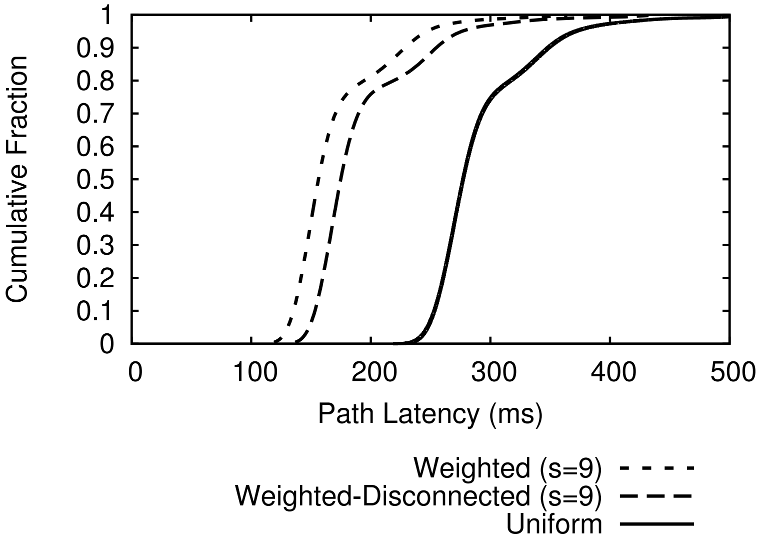


Figure 4.5: The cumulative distribution of e2e path latencies when using variants of the Weighted strategy on the `King` dataset. The performance achieved using `Uniform` is provided for comparison.

4.5.3 Disconnected Endpoints

Finally, initiators can exclude the first (from the initiator) and last (to the responder) links when ranking paths. Rather than weighting paths based on e2e path estimations, the initiator can instead rank paths by the cost of the subsequence R1→R2→R3, where R1, R2, and R3 are the three relays in a candidate path. Since path selection does not consider the link R3→Responder, our modified path algorithm, Weighted-Detached, does not require the responder to maintain a coordinate.

The obvious cost of using Weighted-Detached is that the first and last hops may be expensive, incurring poor performance even though the subsequence R1→R2→R3 may be efficient.

Our experimental evaluation shows that the performance penalty due to WEIGHTED-DETACHED is minimal. Figure 4.5 shows the performance of the vanilla and revised strategies using the `King` topology with $s = 9$. For comparison, the performance achieved using UNIFORM is also plotted. Although the unmodified WEIGHTED algorithm achieves the lowest median e2e path latency (156.3ms), the modified version also achieves significantly lower latencies (174.9ms) than UNIFORM (277.2ms).

As discussed in the following Chapter, WEIGHTED-DETACHED also offers increased protection against certain anonymity attacks.

Chapter 5

Anonymity Analysis

In this chapter, we compare the anonymity properties of link-based and node-based relay selection under various attacker strategies.

5.1 Attacker Model

Prior work utilizes attacker models in which the adversary may supplement the network with additional malicious relays [65]. Link-based path selection is difficult to accurately assess using such models, as the performance and anonymity of anonymous paths depend upon the precise locations of all relays. The network datasets listed in Table 3.2 reflect actual network distances; adding malicious nodes to these datasets requires many assumptions about where such misbehaving hosts might be placed.

Instead, we model an attacker that controls or monitors $f \cdot N$ of an N-node network (i.e., a trace-driven dataset), where $0 \leq f < 1$. We conservatively assume that the adversary has complete network information and may select *a priori* which of the $f \cdot N$ nodes it controls (e.g., those with highest bandwidth). While this is a particularly strong threat model, it enables us to explore the limitations of our techniques by allowing the attacker to select the most "attractive" relays in a realistic network topology. Due to the ease at which an adversary may acquire high performance nodes using a botnet, we view our threat model as conservative, but realistic.

As with the previous two chapters, we utilize the `King`, `PL-ASes`, `PL-Jitter`, `S`^{3}`-BW`, and `Tor-BW` datasets described in Table 3.2.

Following existing literature, we consider an anonymous route to be compromised if and only if the attacker controls its first and last relay [103]. (Recall that in such a situation, the adversary trivially knows the identities of the initiator and responder, and can conduct timing attacks [63, 103] to determine that the first and last relay are the endpoints of the same anonymous path.)

5.2 Node Prevalence: A New Metric for Anonymous Path Selection

To quantitatively compare link- and node-based relay selection, we introduce a new measure of anonymity, *node prevalence*, defined as follows:

Definition 11 (Node prevalence)**.** *The* node prevalence *of a relay X is the expected probability that X is present on an anonymous path, assuming a uniform probability over the a priori sender and receiver anonymity sets.*

Intuitively, the node prevalence of a relay measures how often the relay is expected to be chosen during path selection[1]. Relays with high node prevalences are more attractive to attackers (since the adversary can compromise a path if it controls the path's first and last relays).

[1] Snader *et al.* [103] propose the use of the Gini Coefficient [40] as a summary statistic of the inequality of relay selection. In contrast, node prevalence measures the popularity of a particular node. By calculating the node prevalence of each relay, we can study the worst-case anonymity of a particular path selection technique, which happens when the adversary has under its control the relays with highest node prevalences (i.e., those used most often in anonymous paths).

In practice, there may be a nonuniform probability distribution over the sets of potential initiators and responders. However, since the identities of the communicating parties are not typically known *a priori* to the attacker, node prevalence enables a quantitative comparison of relay selection strategies under this simplifying assumption (in Section 5.3.4, we describe targeted attacks in which this assumption does not hold). Two path selection strategies can be quantitatively compared by comparing their distributions of node prevalences.

In node-based relay selection techniques, high-bandwidth nodes are consistently perceived as attractive to all initiators, leading to relays with high node prevalences. In contrast, the likelihood that a node will be attractive for all paths using link-based approaches is fairly small, since a node's attractiveness is a function of the locations of the initiator, responder, and already chosen relays in the path. As we show below, the ability of link-based relay selection to prevent "hotspots" leads to increased anonymity since a small coalition of malicious relays cannot easily attract a disproportionate amount of traffic.

5.2.1 Node Prevalence for Link-Based Relay Selection

We can empirically derive an estimate of node prevalence for a particular topology by generating multiple paths between all possible pairs of initiators and responders. For each initiator/responder pairing, we compute for each potential relay the fraction of paths for which the relay is a participant. We can then estimate the node prevalence by averaging those fractions for all pairs of initiators and responders (since, by Definition 11, all nodes in the topology are equally likely to be the initiator or responder).

Figure 5.1 plots the *maximum* of all relays' node prevalences – the frequency at which the most popularly chosen node is present in anonymous paths. Even when WEIGHTED is tuned for high performance ($s = 15$), the most popular relay is present

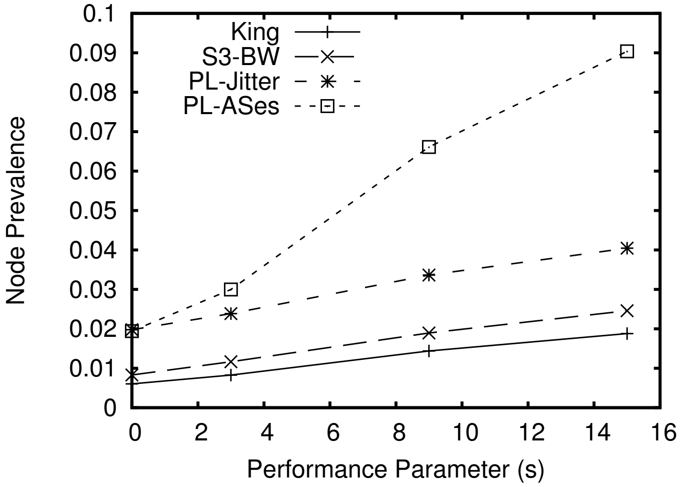


Figure 5.1: The maximum of relays' node prevalences in the `King`, $\texttt{S}^3$`-BW`, `PL-Jitter`, and `PL-ASes` datasets using Weighted.

in less than 5% of paths in the `King`, $\texttt{S}^3$`-BW`, and `PL-Jitter` datasets, and less than 10% of routes using the `PL-ASes` trace. The corresponding performance of the paths is shown in Figure 3.1 through 3.4.

5.2.2 Node Prevalence for Node-Based Relay Selection

In comparison, node-based relay selection yields substantially higher node prevalences. Figure 5.2 shows the maximum node prevalence for the default Tor path selection strategy [27] and Snader and Borisov's proposed refinement [103] using the `Tor-BW` dataset. (Tor's routing algorithm takes no performance parameter and is shown as a straight line.) For both strategies, high bandwidth relays are attractive to all initiators. In particular, the highest bandwidth node is present in 36.9% of all paths produced using the default Tor algorithm. The tunable Snader-Borisov strategy has a modest maximum node prevalence of 2.0% when $s = 3$, but results in much poorer anonymity for greater values of s. When $s = 15$, 79.2% of paths contain

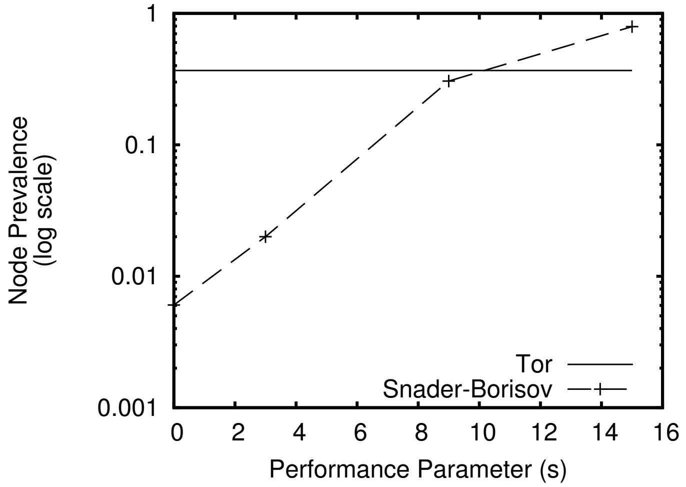
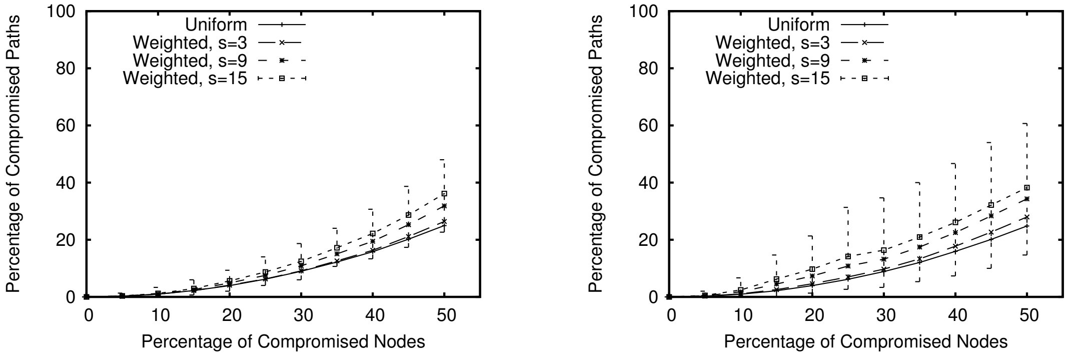


Figure 5.2: Maximum node prevalences of relays in the `Tor-BW` dataset using the default Tor routing algorithm and the Snader-Borisov refinement.

the node with the greatest bandwidth. Although Figure 5.1 and Figure 5.2 cannot be directly compared since they use different underlying topologies and metrics, it is apparent from the figures that while there are no statically-attractive relays as perceived by WEIGHTED, node-based techniques result in hotspots that are present in a large fraction of paths.

5.3 Attack Strategies

We next consider various strategies available to the attacker. As described above, we utilize a conservative attacker model in which the adversary can choose *a priori* which relays he will compromise (up to some fraction f of the network). We further assume that the attacker has complete network knowledge (i.e., pairwise distances) to which to base his decision. Similarly, since our purpose is to evaluate link-based selection strategies, we assume that the initiator knows all network distances with perfect accuracy. We investigate the effect of coordinate-based distance estimations



(a) `King` *(b)* `S`^{3}`-BW`

Figure 5.3: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker uses the `BestLinks` strategy and the initiator uses Weighted with the *(a)* `King` and *(b)* `S`^{3}`-BW` datasets. Points represent the mean value with error bars (for $s = 15$) indicating the 5th and 95th percentiles. Error bars are omitted for $s \neq 15$ for readability. In all cases, the 5th-95th percentile ranges for $s \neq 15$ were less than that for $s = 15$.

on anonymity in Section 5.5.

Unless otherwise indicated, the initiator uses the Weighted relay selection strategy for all attacks. The anonymity offered by Weighted-Detached is explored in Section 5.4.

5.3.1 `BestLinks`: Compromising Attractive Links

In the `BestLinks` strategy, the attacker compromises the endpoints of the most attractive links. Mirroring the behavior of the initiator, the attacker ranks smaller distances more favorably if the metric is latency, jitter, loss, or AS traversals, and views larger distances as more advantageous for bandwidth. Given an ordering of links, the two endpoints of each link are assigned to the attacker until he controls $f \cdot N$ relays.

The effectiveness of the `BestLinks` strategy is depicted in Figure 5.3. The x-axis denotes the fraction of nodes controlled by the attacker (f), while the y-axis plots the

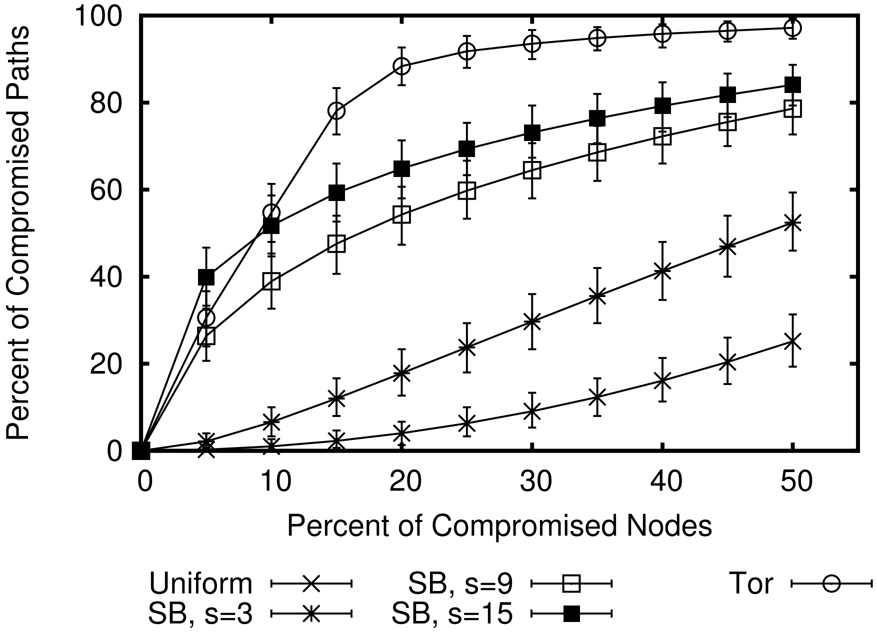


Figure 5.4: The percentage of compromised paths when the attacker uses `BestNodes` and the initiator uses node-based strategies with the `Tor-BW` dataset.

resultant percentage of paths that are compromised. As can be observed from the Figure, Weighted successfully protects most anonymous paths even when the attacker controls 50% of the network. When paths are weighted heavily in favor of performance ($s = 15$) and 30% of the network is controlled by the attacker, only 12.4% of the anonymous paths in the `King` dataset become compromised (Figure 5.3*(a)*). Similarly, for bandwidth (Figure 5.3*(b)*), 16.4% of paths are compromised when 30% of the network is malicious. Results for the `PL-ASes` and `PL-Jitter` datasets are comparable, and are omitted for brevity.

For comparison, Figure 5.4 shows the percentage of compromised paths for node-based selection strategies when the attacker uses the `BestNodes` attacker strategy on the `Tor-BW` dataset. Analogous to `BestLinks`, `BestNodes` ranks nodes according

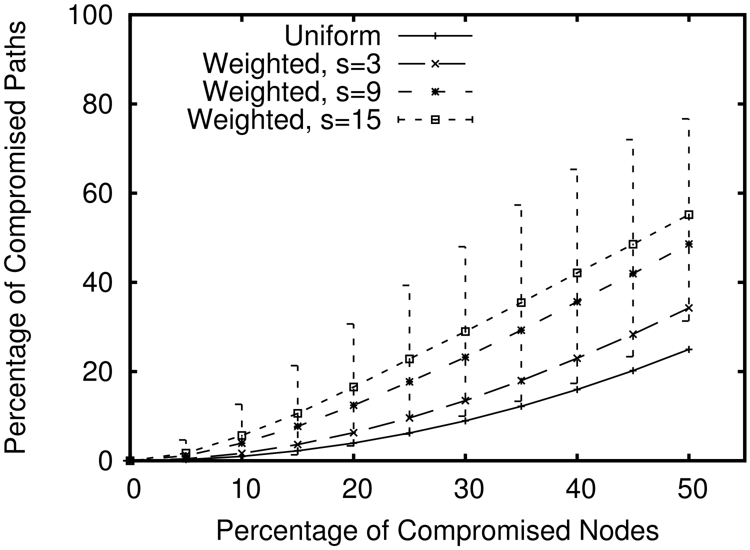


Figure 5.5: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker uses the `MedianDist` strategy and the initiator uses WEIGHTED with the `King` dataset.

to their advertised bandwidths, with the attacker controlling the $f \cdot N$ nodes with greatest bandwidth. `BestNodes` is particularly successful against the default Tor algorithm. When the attacker controls the top 10% of relays, he is able to compromise 54.7% of anonymous paths. The Snader-Borisov ("SB") algorithm fares better for low values of s. However, the strategy becomes vulnerable when performance is more highly valued. An adversary who operates the top 30% of high bandwidth nodes controls 73.1% of paths when $s = 15$.

5.3.2 `MedianDist`: Compromising Nodes with Shortest Median Distances

Alternatively, the attacker may choose the $f \cdot N$ nodes that have the smallest median distance between itself and all other nodes. Intuitively, `MedianDist` locates relays that are likely to be chosen due to their proximity to other relays. Figure 5.5 plots the effectiveness of such a strategy when used with the King dataset. When weighted most heavily in favor of performance ($s = 15$), only 16.5% of paths are compromised when the attacker controls 20% of the network. Results for the remaining link-based topologies are consistent with King and are omitted for brevity. Although MedianDist is more effective than BestLinks, link-based relay selection significantly limits the ability to compromise paths, even against our powerful attacker.

5.3.3 Prevalence: Compromising Nodes with Greatest Node Prevalence

An attacker who employs the Prevalence strategy controls the relays with the highest node prevalences. The Prevalence strategy is near-optimal, since it selects those relays that (by definition) are most often selected (although not necessarily as the first and last relays). Since node prevalences depend not only on the network topology, but also on the initiator's relay selection strategy and its associated parameters, the Prevalence attack is more difficult to perform than the above attack strategies. We include Prevalence in our results to highlight the resilience of link-based relay selection against very powerful adversaries.

Figure 5.6 shows the percentage of compromised paths for the King *(left)* and $\texttt{S}^3$-BW *(right)* datasets. When the initiator attempts to produce paths with low e2e latency ($s = 9$), 24.2% of anonymous paths are compromised when the attacker controls the top 30% relays. Similarly, 24.7% of paths become compromised when 30% of the relays in the $\texttt{S}^3$-BW dataset are under the attacker's control and $s = 9$.

For node-based relay selection, Prevalence is equivalent to BestNodes, since a relay with higher bandwidth always has a higher node prevalence than its lower bandwidth peers. The percentage of compromised paths achieved using BestNodes against the Tor and Snader-Borisov algorithms is plotted in Figure 5.4. When the

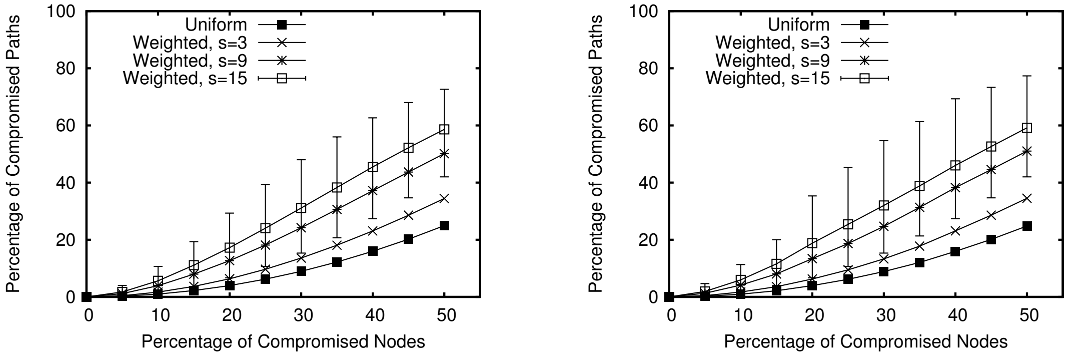


Figure 5.6: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker uses the `Prevalence` strategy and the initiator uses WEIGHTED with the *(left)* `King` and *(right)* $\texttt{S}^3$`-BW` datasets.

adversary controls 30% of the nodes in the `Tor-BW` dataset, he is able to compromise 73.1% of paths when $s = 15$.

5.3.4 `Confirmation`: Determining whether Alice is Communicating with Bob

The previous attacks attempt to compromise arbitrary paths in the anonymous network. In contrast, an attacker may apply the `Confirmation` attack to test whether a fixed pair of nodes (Alice and Bob) is anonymously communicating. Here, the attacker compromises the nearest node (e.g., having smallest RTT) to Alice that has not yet been compromised, and does the same with respect to Bob, and continues compromising nodes in this manner until he has controls $f \cdot N$ nodes. That is, the attacker compromises the nodes that are nearest to Alice and Bob to maximize the probability that he controls the first and last relays in their anonymous path (assuming such a path exists).

The results of using the `Confirmation` strategy against the `King` dataset are shown in Figure 5.7. The figure plots the results of experiments between all pairwise

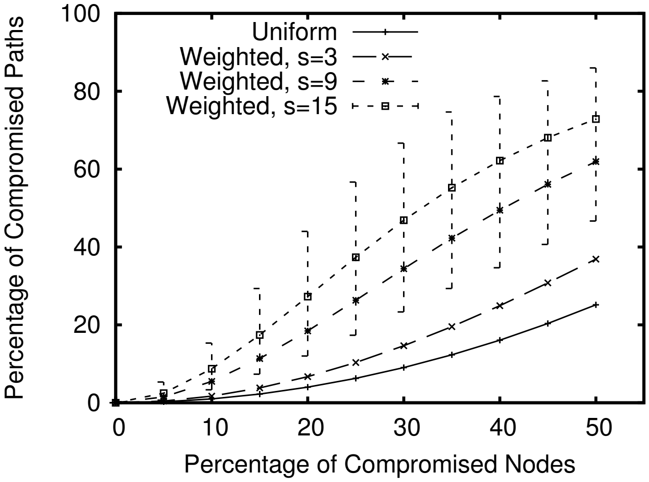


Figure 5.7: The percentage of compromised paths when the attacker uses the `Confirmation` strategy and the initiator uses WEIGHTED with the `King` dataset.

initiators and responders. In each experiment, the attacker compromises the $f \cdot N$ nodes in the manner described above to target the particular initiator and responder pair. When routes are weighted heavily in favor of performance ($s = 9$), an attacker who controls 30% of the network and who can target particular initiator and responder pairs, can verify that they are communicating 34.4% of the time. As discussed in Section 5.4, a slightly modified WEIGHTED strategy better protects against the `Confirmation` attack at the cost of a small degree of performance.

5.3.5 `Cluster`: Joining the Network with a Cluster of Nodes

An attacker may attempt to compromise a large fraction of anonymous paths by joining the anonymity network using multiple nodes from the same LAN. Due to the high bandwidths and low latencies within the LAN, paths composed entirely of malicious nodes from the LAN will have low e2e cost estimates and will be favored by the WEIGHTED algorithm.

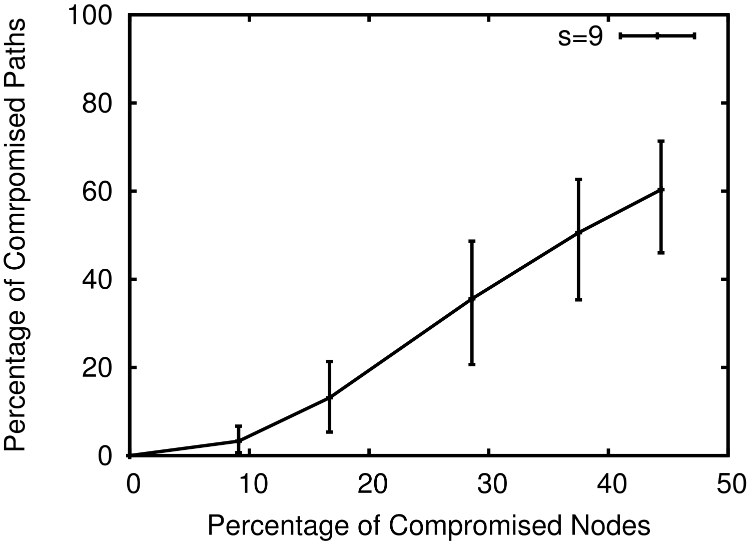


Figure 5.8: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker uses the `Cluster` strategy and the initiator uses WEIGHTED with the `King` dataset using $s = 9$.

Since our experimental datasets do not contain large clusters of similarly located nodes, it was necessary to adapt the attacker model to permit the attacker to insert nodes. To determine the location for the new nodes, we first use the Vivaldi [19] virtual embedding system to assign n-dimensional coordinates to each node in the existing topology such that the Cartesian distance between two nodes' coordinates corresponds to the network distance (e.g., latency) between them. To provide the attacker with a desirable location in the topology, we assign each malicious node a coordinate that is at most 5ms from the centroid of the network. Hence, any two malicious nodes are separated by at most 10ms. Locations from the centroid are randomly chosen according to Muller's uniform hypersphere point generation technique [62]. Network distances between a malicious node and another peer are estimated using the Cartesian distance between the nodes' coordinates.

Figure 5.8 illustrates the efficacy of the `Cluster` attack when the initiator uses the

Weighted algorithm with $s = 9$ on the King dataset. When the attacker controls 28.6% of the network (i.e., he adds 200 nodes to the existing 500 node topology), he compromises just 35.6% of anonymous paths.

It is worth noting that the Cluster attack may be further mitigated by requiring that adjacent nodes in anonymous paths reside in separate autonomous systems or have a minimum latency between them.

5.4 Anonymity Benefits of Weighted-Detached

The Weighted algorithm introduced in Section 3.1 ranks candidate paths by the expected e2e path cost. Unlike node-based relay selection strategies, the e2e path cost includes the links from the initiator to the first relay and from the last relay to the responder. The use of the initiator's and responder's network locations during path selection potentially leaks information about the communication participants.

For example, consider the case in which the adversary controls the middle relay (R2) in a three-relay anonymous path: A→R1→R2→R3→B, where R1, R2, and R3 are relays and A (Alice) and B (Bob) are the respective initiator and responder. Since the attacker controls R2, she trivially knows the identities of the first (R1) and third (R3) relays (since she communicates with them directly), and can therefore compute the cost of routing along the subsequence R1→R2→R3. For all combinations of possible initiators ($\alpha \in N$) and responders ($\beta \in N$), the attacker can generate the set $\mathcal{P}$ of all possible three-participant paths, compute their costs, and determine the ranking $\hat{r}$ of α →R1→R2→R3→ β in $\mathcal{P}$, where $\hat{r}$ is normalized to be in the range $[0, 1)$. From Equation 3.1, the probability that α chose the path α →R1→R2→R3→ β to communicate with β is $f_s(x)/\hat{r}$. The attacker can then use such probabilities to form a nonuniform probability distribution over the sets of potential initiators and responders.

The Weighted-Detached strategy introduced in Section 4.5.3 mitigates such

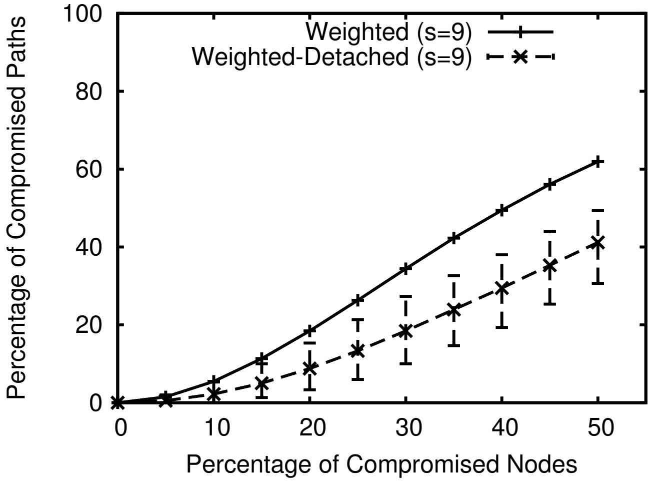


Figure 5.9: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker users the `Confirmation` strategy and the initiator uses variants of Weighted with the `King` dataset.

an attack by excluding the first (from the initiator) and last (to the responder) links when ranking paths. That is, Weighted-Detached disassociates the communication endpoints from path selection, weighing path selection only by the cost of routing between the three relays. An adversary who knows the identities of R1, R2, and R3 cannot infer any information about the initiator and responder.

5.4.1 Resilience to `Confirmation` Attacks

Since the initiator considers neither its nor the responder's locations when producing high performance paths, Weighted-Detached also mitigates the `Confirmation` attack (see Section 5.3.4) in which the attacker attempts to verify that a particular initiator (Alice) is communicating anonymously with a particular responder (Bob). Although a strong adversary can compromise the relays that are closest to Alice and Bob, such an attack strategy is not nearly as effective against Weighted-Detached as it is against Weighted.

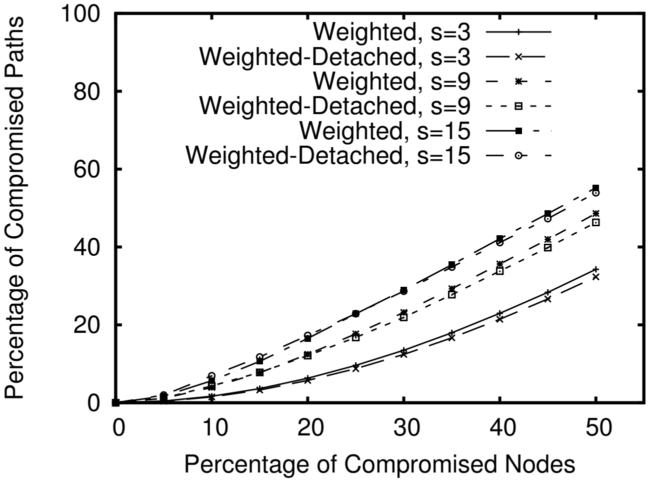


Figure 5.10: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker users the `MedianDist` strategy and the initiator uses variants of Weighted with the `King` dataset.

Figure 5.9 compares the resilience to the `Confirmation` attack using the two path selection strategies. Since the positions of the initiator and responder do not influence relay selection when Weighted-Detached is used, the attacker's strategy is less effective. For example, when 30% of nodes are malicious, the attacker compromises 34.4% of paths when the initiator uses the unmodified Weighted technique and only 18.5% against Weighted-Detached.

5.4.2 Resilience to `MedianDist` and `Prevalence` Attacks

Unsurprisingly, the use of Weighted-Detached rather than Weighted does not significantly impact the effectiveness of the other attack strategies described in Section 5.3, since these attacks do not consider the locations of the initiator and responder. To illustrate, Figure 5.10 plots the percentage of compromised paths when the attacker utilizes the `MedianDist` strategy and the initiator uses Weighted or

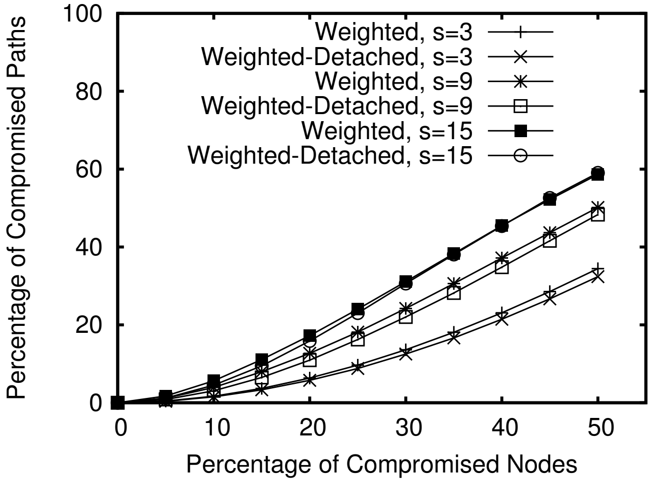


Figure 5.11: The percentage of compromised paths as a function of the fraction of compromised nodes when the attacker users the `Prevalence` strategy and the initiator uses variants of WEIGHTED with the `King` dataset.

WEIGHTED-DETACHED. The `MedianDist` strategy is nearly equally (in)effective against both relay selection techniques. For example, when the attacker controls 30% of the network and the initiator sets $s = 15$, 29.0% of all WEIGHTED paths and 28.6% of all WEIGHTED-DETACHED paths are compromised. Similarly, Figure 5.11 compares the effectiveness of the `Prevalence` attack against WEIGHTED and WEIGHTED-DETACHED. As before, the performance of the attack does not increase when WEIGHTED-DETACHED is used. When 30% of the network is malicious and $s = 15$, the corresponding path compromise rates are 31.1% and 30.6% for WEIGHTED and WEIGHTED-DETACHED, respectively.

5.4.3 Preventing the Predecessor Attack

An anonymized connection between an initiator and responder is often reset due to node churn, requiring it to be reconstructed using different relays [116]. The adversary can conduct a *predecessor attack* to discover the initiator by counting the number of times each relay precedes the attackers' relays in the anonymous path [83, 116]. Since the initiator is always present in such circuits, it will have a higher count than the relays that are chosen randomly whenever the circuit is rebuilt.

Tor mitigates the predecessor attack by using a small number of fixed entry nodes called *guards* [27]. Link-based path selection is equally vulnerable to the predecessor attack, but may also be defended using guards. Guards must be chosen carefully since their locations affect the performance of a path. However, as described in Section 4.5.3 (see, in particular, Figure 4.5), link-based routing produces high performance paths even if the first hop (connecting the initiator to the guard node) is not considered by the path selection algorithm. Link-based routing may therefore adopt the same mitigation strategy as Tor [28]; namely, the initiator selects a relay (having a long uptime) to act as its entry guard for all anonymous paths.

5.5 The Impact of Coordinate Systems on Anonymity

Coordinate systems linearize the amount of information required to store pairwise distances by mapping each node to n-dimensional coordinates. Due to network triangle inequality violations (TIVs) that cannot be represented in Euclidean space, coordinate systems do not perfectly predict distances. As shown in Figures 4.2 through 4.4, the use of coordinate systems imposes a modest decrease in path performance. In this section, we consider the effects of using coordinate systems on anonymity.

Coordinate systems effectively transform a topology of pairwise distances in which TIVs may occur into a *similar* topology in which no TIVs exist. The lack of TIVs eliminates "wormhole" effects in the network in which it is advantageous to route

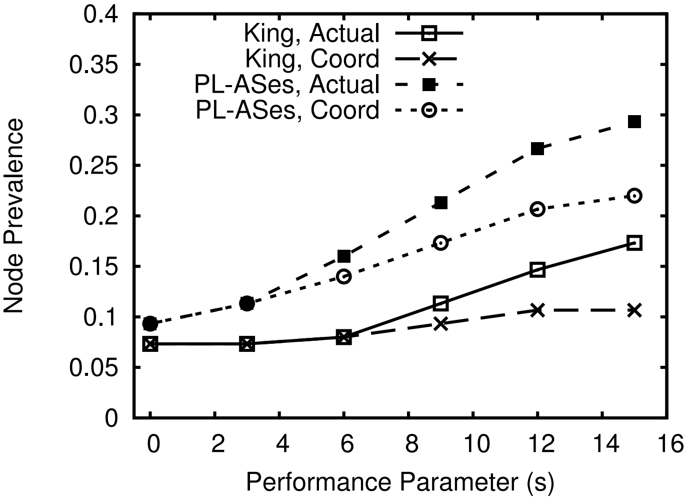


Figure 5.12: The *maximum* node prevalences for actual and Coordinate-based ("Coord") distances for the King, PL-ASes, and PL-Jitter datasets.

through a particular relay rather than connect directly to the receiver.[2] Consider, for example, the case in which an initiator A has very low latency to a relay R. Although the actual e2e performance of A's paths may be improved by consistently routing through R (due to TIVs), the use of Euclidean virtual coordinates likely overestimates the cost of $A \rightarrow R$ since the cost of $A \rightarrow R \rightarrow B$ can never be less than $A \rightarrow B$ for any responder B.

Such behavior was observed when embedding the King and PL-ASes topologies using the Vivaldi coordinate system. Figure 5.12 plots the *maximum* of all relays' node prevalences for WEIGHTED when actual network distances are used to estimate path performance ("Actual") and when coordinate estimates are utilized ("Coord"). As can be observed from the Figure, the maximum node prevalence – the frequency in which the most popularly chosen relay is present in anonymous paths – *decreases*

[2] In particular, we notice that such "wormholes" are ubiquitous on PlanetLab [74] due to the mixed availability of Internet and Internet2 connectivity among nodes. When Internet2 connections are not available between two nodes, lower latency can often be achieved by relaying traffic via multihop paths in which the largest geographic distance is covered by an Internet2 link.

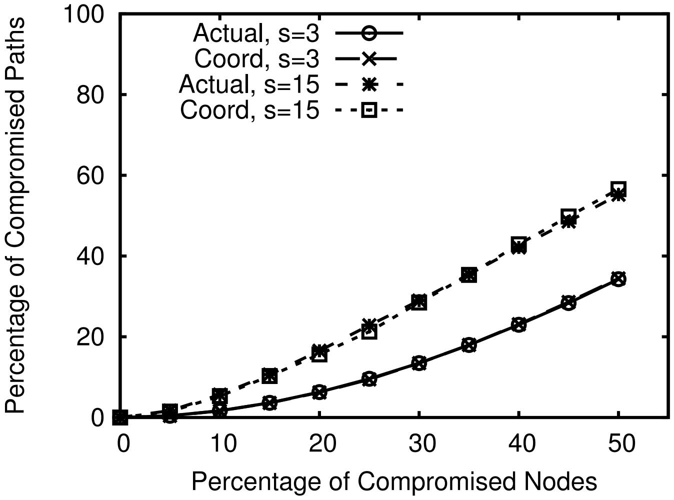


Figure 5.13: The percentage of compromised paths on the King dataset when the attacker uses the MedianDist strategy. "Actual" denotes results obtained using actual network distances. "Coord" reflects performance when initiators estimate distances using the coordinate embedding system.

when coordinate systems are utilized.[3] Hence, although coordinate systems incur a modest decrease in communicating performance, they provide slightly greater anonymity by diversifying anonymous paths.

To illustrate, Figure 5.13 shows the percentage of compromised paths using the King dataset when the attacker applies the MedianDist strategy. The figure compares performance results obtained using network distances ("Actual") to estimations based on the coordinate system ("Coord"). As can be observed from the Figure, the use of virtual coordinates to estimate distance does not decrease anonymity. For example, when $s = 15$ and actual distances are used, an attacker who controls 30% of the network can compromise 29.0% of paths. In comparison, the same attacker can compromise 28.5% of paths when virtual coordinates are used in place of

[3] The difference in maximum node prevalences between topologies is primarily due to the differing network sizes. Smaller networks produce higher node prevalences since there are fewer relays that may be selected.

actual distances. Analogous results were drawn from comparing the performance of the `BestLinks` and `Prevalence` attacks using actual and virtual distances.

Chapter 6

Application-Aware Anonymity (A^3)

In this chapter, we introduce *Application-Aware Anonymous Routing* (A^3), a flexible network overlay that enables senders to intelligently select anonymous relays using accurate network distance estimations. A^3 is a fully-distributed peer-to-peer (p2p) anonymity system in which all participating nodes serve as potential relays. The core of the A^3 routing algorithm is the WEIGHTED relay selection strategy introduced in Section 3.1, enabling applications to tradeoff between anonymity and performance.

Unlike existing anonymity systems that depend on central authorities or directories (e.g., Tor [28]), A^3 does not rely on *a priori* trusted nodes or third-party authorities. This lack of centralization enables A^3 to scale to potentially hundreds of thousands of nodes and offer anonymity that does not depend on the trustworthiness of select nodes or services.

In the following sections, we describe A^3 and evaluate its anonymity properties and performance under both simulation and a testbed deployment on PlanetLab.

6.1 Design Goals, Assumptions, and Limitations

A^3 has three principal design goals: (i) to permit flexible high performance anonymous routing, (ii) to protect the identities of the anonymous communicants, and (iii) to avoid reliance on *a priori* trusted nodes or services. In Chapter 3, we described link-based relay selection techniques that achieve the first two goals by using coordinate-based distance estimations to bias relay selection in favor of high performing paths. In this section, we justify our decentralized trust model and discuss its implications to security and anonymity.

A^3 requires no central authorities and further assumes no pre-shared knowledge. Anonymity networks that rely on centralized servers (e.g., directories) implicitly trust such services to behave correctly and honestly. If these centralized services are malicious, or if they are coerced or compromised, then the network provides no anonymity to any of its users. A corrupt directory server could, for example, answer all queries with the identities of malicious peers. Even if the directory servers are trustworthy, the anonymity network's centralized trust model presents a valuable target of attack. For example, although operated by trustworthy individuals, nearly half of Tor's directory servers were found to be vulnerable to the Debian OpenSSL key generation bug [26]. Since the vulnerable directory servers generated private keys using just 17 bits of entropy [23], an attacker could easily brute-force their keys to forge "signed" messages purportedly from the directory server.

A fully distributed system such as A^3 has no central point of trust or of failure. Of course, individual peers may behave dishonestly. However, the lack of centralization removes attractive targets for attack; the effect of compromising a node is localized to only a small fraction of the network. Moreover, all of A^3's distributed components are fortified by various distributed security schemes (described later in this chapter and in Chapter 7), limiting the effects of even large-scale attacks.

However, A^3's lack of specialized nodes or pre-shared knowledge introduces some fundamental limitations – most importantly, the impossibility of secure key exchange.

Without trusted authorities, an active eavesdropper (Eve) can conduct a man-in-the-middle (MitM) attack between any two nodes, thwarting any attempted key exchange. At the extreme, Eve could intercept all of Alice's traffic and emulate a fictitious network unbeknowst to Alice. Hence, there is a tradeoff between anonymity systems that are fully decentralized with no pre-shared keys and services that depend on *a priori* pre-shared information such as certificate authorities. For the latter, MitM attacks become more difficult as keys can be securely exchanged using authenticated key agreement protocols. However, as discussed above, the anonymity of these systems depends on these central authorities.

The distributed A^3 architecture, as presented in this chapter, does not protect against active adversaries that conduct MitM attacks at network *edges* (i.e., locations near communication participants). We opt instead to introduce a fully decentralized architecture that, at the cost of enabling the aforementioned vulnerability, focuses on protecting the anonymity network as a whole by eliminating central points of trust. We acknowledge that such a tradeoff is controversial and may be appropriate only for certain applications. For example, A^3 may be particularly well-suited for applications in which *receiver anonymity* (the inability to identify the receiver of an intercepted message) is more paramount than *sender anonymity* (the inability to determine its sender).

It is worth emphasizing that although A^3 is framed as a fully decentralized anonymity network requiring no *a priori* shared secrets, A^3 is not inherently incompatible with a public key infrastructure (PKI). At the cost of making anonymity dependent on the security of the certificate authorities, a PKI could be straightforwardly layered on top of A^3 (e.g., by requiring all messages to be signed by their senders).

6.2 Performance Advantages of a Fully Distributed Architecture

A^3 is a p2p system, and hence any participating node can function as an initiator (the sender of anonymous content), a participant (an intermediary relay), a responder (the initiator's target of communication), or any combination of the above. We argue that A^3's fully distributed architecture permits greater performance than more centralized approaches in which traffic is forwarded through a small collection of mostly fixed relays. As shown in Figure 1.3, Tor suffers from poor performance, likely because of the relatively few number of relay nodes (≈1500) compared to the number of clients (prior work estimates the number of Tor users to be between 100,000 and one million [60]). Overburdened relays suffer from congestion, causing poor e2e path performance [29]. In contrast, a fully distributed anonymity network in which any node may act as a relay better equalizes the number of clients and relays, thus eliminating a large source of congestion.

Furthermore, p2p architectures appear better suited for traffic that traverses anonymity networks. Existing studies have shown that BitTorrent, a p2p file sharing network, accounts for slightly more than 40% of bandwidth traversing the Tor network [60]. P2p anonymity networks are better matched at handling traffic from p2p file sharing applications.

Finally, it has been noted that although Tor clients are geographically diverse, the vast majority of bandwidth traversing Tor resides in a small fraction of countries [60]. For example, 45% of Tor bandwidth is local to Germany [60]. Autonomous systems located in such jurisdictions are therefore more likely to be able to observe traffic traversing the first and last participants of anonymous paths, and can apply the timing attacks described in Section 2.4.4 to identify the communicating parties. By encouraging clients to serve also as potential relays, A^3 inherently causes the distribution of clients to match that of relays.

Although we believe that a fully distributed architecture is best suited for a general purpose, performance tunable anonymity network, it is important to note that such an infrastructure also has certain disadvantages. Mix-based anonymity systems shuffle streams at relays to aggrevate an observer's ability to correlate inbound and outbound traffic [14, 82, 72, 9, 35]. By design, A^3 reduces the number of streams traversing a particular router, and cannot therefore readily apply mixing. (Note that more centralized low-latency anonymity systems, e.g. Tor, choose not to mix traffic in favor of providing paths with low latencies [28].) Additionally, larger anonymity networks are *more* vulnerable to counting attacks (see Section 2.4.1). However, exploiting counting attacks typically requires an adversary who has a global view of the network. A highly distributed p2p architecture makes acquiring such a wide network view much more arduous.

6.3 System Architecture

A^3 is designed as an Internet-scale system in which we envision supporting up to millions of simultaneous users. To avoid performance bottlenecks, A^3 utilizes highly scalable distributed services.

At a high-level, A^3 is divided into four core components:

- A **distributed directory service** provides efficient node discovery and message delivery operations. Our A^3 implementation utilizes distributed hash tables (DHTs) [6] due to their support for scalable lookups.

- An **embedded coordinate system** (e.g., Vivaldi [19], PIC [16], NPS [66], etc.) maps nodes to n-dimensional Euclidean coordinates such that the Euclidean distance between any two nodes corresponds to a *metric of interest* (e.g., latency, number of AS traversals, etc.) between the two nodes. Coordinate systems are protected from manipulation by the Veracity security mechanism described in Chapter 7.

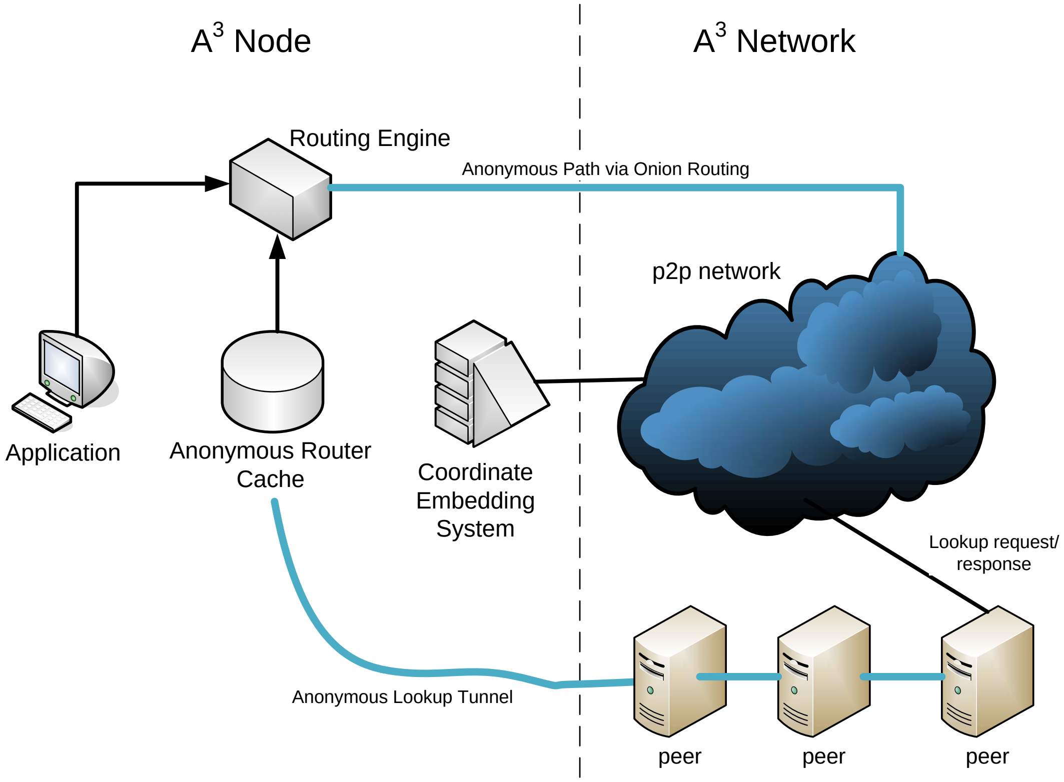


Figure 6.1: A^3 system architecture.

- Each peer uses **anonymous lookup tunnels** to anonymously locate potential relays.

- Finally, a **routing engine** serves as an interface between the application, a node's anonymous lookup tunnel, and the coordinate embedding system. The routing engine allows applications to use information obtained from anonymous lookups to form high performance anonymous circuits.

Figure 6.1 illustrates A^3's high level architecture. Each A^3 peer participates in one or more coordinate embedding systems, with each system representing a separate network metric (e.g., latency, jitter, etc.)[1]. Corresponding to each coordinate system, each node maintains a coordinate for each network metric. Periodically,

[1] Alternatively, a single more highly dimensional coordinate embedding system may be used in which different dimensions are reserved for particular metrics.

nodes use the distributed directory service to locate a random peer in the network, and then conduct the underlying coordinate system protocol with that peer in order to update the local coordinate. To protect the truthfulness of advertised coordinates and measurement probes, Veracity (see Chapter 7) is layered on top of the coordinate embedding systems.

Each node maintains an *anonymous router cache*, a collection of peers' network addresses, coordinates, and public keys. The anonymous router cache is used as a repository of peer information to construct overlay paths. To populate its anonymous router cache, a node instantiates one or more anonymous lookup tunnels – onion circuits that provide sender anonymity – to random nodes in the network. The local node anonymously relays lookup requests to the endpoint of the tunnel. Requests are resolved by the tunnel's endpoint and are subsequently relayed backwards through the lookup tunnel where they are eventually added to the node's anonymous router cache.

Selecting relays from the entries in the local node's anonymous router cache, the routing engine produces high performance onion routes using the WEIGHTED relay selection algorithm introduced in Section 3.1.

Below, we describe A^3's operation in more detail.

6.3.1 Distributed Directory Service

A^3 makes use of a distributed directory service to locate candidate nodes during path selection. In addition to scalability, the directory service must meet the following requirements:

- **Hashed node identifiers:** Each A^3 node must have a globally unique identifier (GUID), computed by taking a SHA-1 hash of its IP address and port. Since peers can quickly verify whether or not a GUID matches with an IP address, this prevents a node from choosing an arbitrary GUID. The cryptographic hash function causes GUIDs to be distributed roughly uniformly over the integers $[0, 2^{160})$.

- **Consistent hashing:** Given an integer $r \in [0, 2^{160})$, the directory service should be capable of efficiently delivering a message addressed to r to the node whose GUID most closely[2] maps to r. This mapping must be consistent among all nodes.

  The only functional requirement of the distributed directory service is that it supports the `deliver` API function:

  `deliver`(g, m) – deliver message m to node whose GUID is closest to g

Distributed directory services that meet the above two requirements enable nodes to locate random peers in the network. To find a peer, a node simply addresses a `deliver` message to an identifier g chosen uniformly at random from $[0, 2^{160})$.

In our implementation (see Section 6.4), we utilize distributed hash tables (DHTs) as our distributed directory service. DHTs meet the above requirements and offer efficient $O(\log N)$ queries (where N is the number of online nodes in the network). Furthermore, DHTs have been shown to be robust to the high levels of churn that are present in many p2p systems [85, 7].

Attacks against the Distributed Messaging Service The performance of A^3 depends on the reliability of the underlying distributed directory service used to route messages through the overlay network. DHTs, in particular, are known to be vulnerable to several classes of insider attack [12, 110]. Unlike the traditional Internet routing infrastructure in which backbone routers require special resources (e.g., connectivity and money), messages in an overlay are routed between peers (who require only Internet connectivity). There are no restrictions as to who can join the

[2] The distributed directory service is responsible for suitably defining *closeness*. For example, the Chord DHT [106] models a circular key space in which a node i with GUID g_i is responsible for all keys in the range $g_j + 1 ... g_i$ (modulo the size of the keyspace) where g_j is the GUID belonging to an adjacent node j. Other DHTs [87, 81, 59, 85] have similar definitions.

network, and therefore a cabal of malicious users can participate and interfere with messages that pass through nodes under their control.

Malicious nodes can conduct *Sybil attacks* to increase their influence by registering multiple identities in the network [31], *eclipse attacks* in which they falsify routing update messages to corrupt honest nodes' routing tables [102], and *routing attacks* in which they inject spurious responses to messages that cross their paths [12]. Fortunately, well-studied techniques exist that defend DHTs against such attacks [25, 22, 11, 36, 12, 5]. We describe DHT defenses that are compatible with A^3's design below.

Sybil attack countermeasures that are compatible with a decentralized architecture include *distributed registration* in which *registration nodes*, computed using iterative hashing of a new node's IP address, vote on whether the new node can join the system based on the number of similar requests it has received from the same IP address [25]. Alternatively, Danezis *et al.* propose using *bootstrap graphs* that capture the relationships between joining nodes and the nodes through which they join to construct trust profiles [22]. Finally, Borisov suggests the use of cryptographic puzzles (e.g., finding a string in which the last p bits of a cryptographic hash are zero) to increase the cost of joining the network [11].

There are also several security techniques that mitigate eclipse and routing attacks. For example, the S-Chord system proposed by Fiat *et al.* organizes the network into *swarms* based on GUIDs [36]. Lookups are relayed between swarms, and are only forwarded if the lookup was sent from a majority of the members of the previous swarm. S-Chord is resilient to attacks in which the adversary controls $(1/4 - \epsilon_0)z$ nodes, where $\epsilon_0 > 0$, z is the minimum number of nodes in the network at any given time, k is a tunable parameter, and the number of honest nodes is less than or equal to z^k. Castro *et al.* propose the use of *redundant routing* in which queries are sent via diverse paths [12]. Routing will reach the intended recipient if all nodes on at least one path are honest. Sanchez *et al.* improve the redundant routing technique in their Cyclone system [5], showing that 85% of requests were correctly delivered when attackers controlled 30% of a 1024 node network and nodes sent messages using eight redundant paths.

The impact of utilizing the above secure routing techniques are minimal. All of the above approaches operate below A^3's protocols and do not affect A^3's operation.

6.3.2 Embedded Coordinate System

The coordinate embedding component of A^3 operates as a background process on each peer, periodically updating the local node's coordinates. Unlike other coordinate system implementations [19], A^3 does not use fixed neighborsets to exchange coordinates. Instead, A^3 nodes periodically locate a random peer by sending a `deliver` request to a random GUID. Once a peer is located, the local node measures some link characteristic (e.g., RTT, bandwidth, etc.) between itself and the peer, and uses the measured distance and the peer's coordinates to update its own coordinates. The use of random peers rather than fixed neighbors enables A^3 to better tolerate churn (since fixed neighbors may leave the network) and encourages better network diversity (since whenever a peer is chosen, it is chosen approximately uniformly at random from all A^3 nodes).

A^3 is compatible with any decentralized coordinate embedding system in which local nodes update their coordinates by making empirical measurements to other peers. Since Vivaldi [19] has received the most study recently from the p2p community [44, 96] and because it has been implemented in both the p2psim [39] peer-to-peer simulator and the Bamboo DHT [7], our A^3 implementation uses Vivaldi to embed network distances.

Unfortunately, the distributed nature of coordinate systems make them particularly vulnerable to insider manipulation. To illustrate, recent studies [45] on Vivaldi have shown that when 30% of nodes lie about their coordinates, Vivaldi's accuracy decreases by a factor of five. When attackers collude, even 5% malicious nodes have a sizable impact on the system's accuracy.

Although a number of techniques have been proposed to protect coordinate embedding systems [89, 16, 44], all existing work of which we are familiar depends on either pre-shared secrets (e.g., the identities of special trusted nodes) or central authorities. In Chapter 7, we describe *Veracity*, a fully distributed protection mechanism for logical coordinate systems that is well-suited for A^3.

6.3.3 Anonymous Lookup Tunnels

The routing engine, described in detail in the following section, uses the WEIGHTED relay selection algorithm to instantiate high performance anonymous paths. In A^3, relays are selected from the entries stored in a node's anonymous router cache.

Each node populates its anonymous router cache by periodically retrieving the coordinate, network address, and public key of a random peer. Such lookups must be anonymized, else a locally-positioned observer can record which relays are known to a particular node. This information can be conveyed to other eavesdroppers, who can then deduce the initiator of an anonymous communication by observing the relays that are present in an anonymous path.

Rejected: Recursive DHT Lookups Distributed hash tables (DHTs) support both *iterative* and *recursive* lookups. In iterative lookups, the requesting node directly contacts a peer to locate the peer's neighbor that is closer to the requested key. The requesting node resends her query to the neighbor, and repeats the procedure until it locates the node that is closest to the requested key. In contrast, in recursive lookups, the requesting node contacts only one of its neighbors; the request is then forwarded from peer to peer until reaching the closest node. The reply traverses the recursive path in the opposite direction until it reaches the requester.

Recursive lookups therefore provide a modicum of anonymity. A peer who receives a recursive lookup cannot ascertain whether the request originated from the node that sent it the request or from a node further upstream. However, as has been noted by O'Donnell [67] and Borisov [10], DHTs leak information about the initiators of recursive lookups by forwarding queries to nodes that are closer to the destination according to a distance metric. Hence, a node can immediately rule out any peer that is closer to the origin than itself as the possible initiator of the request. A^3 seeks to achieve a stronger form of anonymity than that provided by recursive lookups.

Our Approach: Onion Tunnels Similar to the Sybil-attack countermeasure introduced by Danezis *et al.* [22], our onion-based anonymous lookup technique exploits the trust relationship embedded in the introduction graph (sometimes called the bootstrap graph) of the network. To join A^3, a node (Alice) contacts an already joined node (for clarity, Alice's "friend") to gain membership. If the friend is malicious, then it can provide false information to Alice, causing Alice's overlay routing table to entirely consist of malicious peers. At the extreme, a malicious friend can simulate an entire network in an attempt to convince Alice that her communications are relayed through a series of peer relays. More generally, in any p2p system that does not rely on *a priori* shared knowledge (for example, the public key of a certificate authority), Alice must trust the friend through which she joins the network.

Since her friend must be honest for Alice to achieve any anonymity, we rely on the friend to bootstrap the join process.[3] At a high level, our anonymous lookup tunnel approach works as follows: Alice uses her friend to discover random peers. Alice's friend locates random peers by querying for random keys, using secure DHT lookup techniques (for example, those listed in Section 6.3.1) to mitigate DHT routing attacks. The friend relays the network addresses and public keys of located peers to Alice, where they are stored in Alice's *directory cache.* To populate her *anonymous*

[3] Although Alice places *a priori* trust in her friend, A^3 does not require that all peers trust specialized and fixed trusted nodes. With the exception of the first joining A^3 node, each peer may utilize a separate friend to join the network. And, as argued above, a node which joins *any* fully distributed system by contacting a peer for membership places implicit trust in that peer.

router cache, Alice uses entries from her directory cache to form onion routes, the endpoints of which locate peer nodes on Alice's behalf without revealing her identity. It should be emphasized that Alice's anonymous router cache (used by the routing engine to form high performance anonymous paths) and her directory cache (used to build onion paths to locate peers to populate the anonymous router cache) are independent (although not necessarily disjoint).

We next describe our onion-based anonymous DHT lookup mechanism in more detail.

When Alice joins A^3 via her friend, Alice requests that her friend fetch the network addresses and public keys of w random A^3 nodes. The friend uses secure DHT lookup techniques to locate the w peers, and forwards the results back to Alice, who then adds the results to her directory cache. Under the assumptions that Alice's friend is honest and that secure DHT techniques effectively mitigate routing attacks, the directory cache consists of w actual (but not necessarily honest) A^3 participants. w should be sufficiently large (on the order of 100 nodes) to enable Alice to form multiple onion routes.

The members of Alice's directory cache are used as potential onion relays, the endpoints of which resolve DHT lookup requests on Alice's behalf. Alice maintains z onion routes, consisting of (and terminating with) members of her directory cache. When onion circuits break due to node churn or network failures, Alice replaces the broken circuit with a new onion route.

To populate the anonymous router cache, Alice chooses a key r' uniformly at random from the GUID keyspace and forwards a DHT lookup request for r' to the endpoints of her z anonymous onion circuits. The endpoint of each circuit uses secure DHT lookup techniques to resolve r' and relays the results backwards through the onion circuit. Since onion routes preserve sender anonymity, the endpoints that resolve DHT lookup requests cannot identify Alice as the requests' originator. (In fact, since Alice never directly contacts the w nodes in her directory cache, the endpoints have little information that they can use to identify Alice as a likely requester.) Alice uses a consensus-like approach to ensure that endpoints do not inject spurious responses to her queries. Alice adds the result of resolving r' to the anonymous router cache iff $\lceil \frac{z}{2} \rceil$ endpoints return the same response.

6.3.4 Routing Engine

The *routing engine* generates high performance anonymous paths using the relays stored in the anonymous router cache. To populate the cache, the routing engine uses an anonymous lookup tunnel to fetch the network addresses, public keys, and coordinates of random peers. The number of entries in the cache is determined by the rate at which the routing engine seeks out peers and by the timeout period after which old entries are removed. If the expiration time for cache entries is too long, entries in the cache risk becoming stale due to dynamic network conditions. In contrast, if the expiration time is too short, the number of entries in the anonymous router cache may be insufficient to allow the formation of high performing circuits. However, the routing engine can dynamically adjust both the rate at which it makes requests as well as the cache expiration time. By carefully adapting its parameters, the routing engine can ensure that a fresh cache of sufficient size is consistently maintained.

To construct high performance anonymous paths, the routing engine uses the WEIGHTED algorithm introduced in Section 3.1. WEIGHTED outputs relay paths using node information stored in the anonymous router cache. If the responder does not participate in A^3 or if the initiator does not know the responder's coordinate (i.e., the responder's information is not present in the anonymous router cache), then the initiator can either use the WEIGHTED-DETACHED relay selection strategy or rely on the closest relay services or AS path estimation techniques described in Section 4.5.

Once WEIGHTED (or WEIGHTED-DETACHED) has selected a three-relay anonymous path, the routing engine generates session keys and encodes them within an onion, multiply encrypted using the relays' public keys [82]. The routing engine then disseminates session keys to the selected relays by transmitting the onion, informing each relay of its previous and next hops. As with traditional onion routing, data is multiply encrypted using the symmetric session keys, with each relay decrypting the outermost layer of the communication before forwarding it to the next hop, eventually reaching the intended recipient. Backwards traffic traverses the anonymous circuit in the reverse direction, with each relay adding a layer of encryption using its symmetric keys.

6.4 Implementation and Evaluation

We now proceed to describe our implementation of A^3 and evaluate its performance under various configurations and attacker scenarios.

6.4.1 Implementation

A^3 is implemented as an add-on to the Bamboo DHT [7]. Bamboo was chosen for its resilience to high levels of node churn [85], its open-source code base, its inclusion of the Vivaldi coordinate embedding system [19], and its ability to run either in simulation mode or over an actual network using the same code base (with the exception of the simulator's virtualized network layer). Additionally, Bamboo uses the SEDA [113] event-driven programming paradigm designed for highly scalable Internet services.

A^3 utilizes Vivaldi as the underlying coordinate system, using a five dimensional coordinate plane. Our implementation constructs anonymous paths using the WEIGHTED or UNIFORM (i.e., WEIGHTED with $s = 0$) path selection strategy, as configured by the client. As with traditional onion routing [82], A^3 disseminates session key information to path participants during the onion setup phase and multiply encrypts messages during the data transmission phase. Onion headers are encrypted under the participants' public keys using RSA and data sent via onion paths are multiply encrypted under AES.

Our setup is as follows. The coordinate embedding process on each node attempts to find a random peer to update its coordinate every five seconds. Additionally, clients maintain at least 100 entries in their anonymous router caches. Entries remain in the cache for 15 minutes. If the anonymous cache has fewer than 100 entries, the client fetches the network addresses, public keys, and coordinates of random peers, adding the results to the anonymous router cache, until the cache contains at least 100 entries. As a workload generator, a background process on each node issues requests for anonymous paths every 15 minutes. The routing engine constructs an anonymous path using the Uniform or Weighted path strategy (depending on its configuration) that terminates at a responder selected uniformly at random from the anonymous router cache. All anonymous paths contain three participants, excluding the initiator and responder. To measure the RTT, jitter, and loss rates of anonymous routes, application-layer "ping" messages are sent via the encrypted tunnel towards the responder. All measurements include both communication and processing overhead (i.e., applying cryptographic operations and serving messages at the application layer).

6.4.2 Simulation Results

The preliminary experiments described in this section were conducted using Bamboo's simulation mode, enabling us to conduct large-scale experiments using network-driven traces. Simulation results were averaged over five runs.

Path Estimation Accuracy The ability to reliably estimate the e2e characteristics of potential anonymous paths is influenced by the accuracy of the underlying coordinate system as well as the freshness of coordinate information in nodes' anonymous router caches. Inaccurate coordinate systems or stale cache information will lead to suboptimal relay selection, since the cost estimates of candidate paths will not accurately reflect their true costs.

This section aims to demonstrate that the A^3 architecture enables nodes to accurately estimate path costs.

Figure 6.2 *(top)* plots the cumulative distribution of *path estimation errors* – the difference between the estimated and actual path costs – for the `King` dataset. We make two observations from the Figure. First, the path estimation errors are small, indicating that cached coordinates are sufficiently accurate and fresh to accurately estimate path costs. For example, the median path estimation error for the `Uniform` selection strategy is only 16.5ms (recall that anonymous routes comprise four hops and, as shown in Figure 3.2, the median latency in the `King` dataset is 63.0ms). Second, we observe that the security parameter (s) does not significantly affect estimation accuracy. The median path estimation errors for $s = 3$ and $s = 15$ using the WEIGHTED relay selection algorithm are 14.5ms and 12.6ms, respectively. The slightly smaller estimation error for $s = 15$ is expected, given that weighting relay selection more in favor of performance yields paths with lower e2e latencies.

Similar conclusions were obtained using the `PL-ASes` and `PL-Jitter` datasets. The cumulative distributions of path estimation errors for the `PL-ASes` and `PL-Jitter` datasets are plotted in Figures 6.2 *(middle)* and *(bottom)*, respectively.

Path Performance The e2e latencies for anonymous paths are shown in Figure 6.3 *(top)*. Unsurprisingly, UNIFORM exhibited the worst performance as relays were selected uniformly at random (without replacement) from the anonymous router

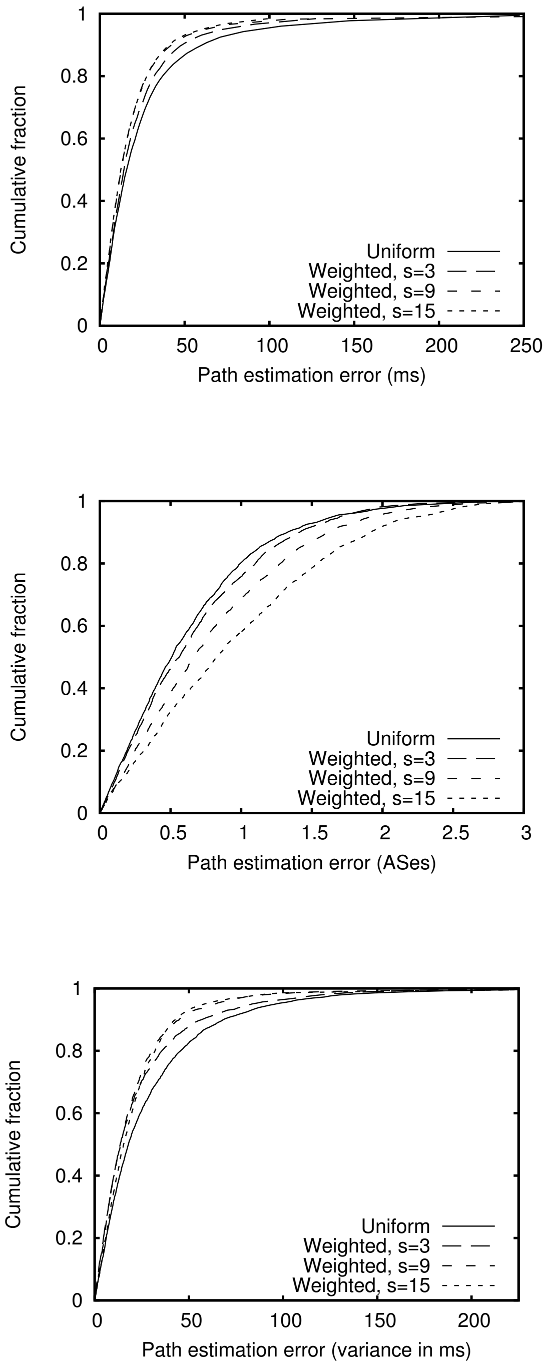


Figure 6.2: Cumulative distribution of the path estimation errors for the `King` *(top)*, `PL-ASes` *(middle)*, and `PL-Jitter` *(bottom)* datasets

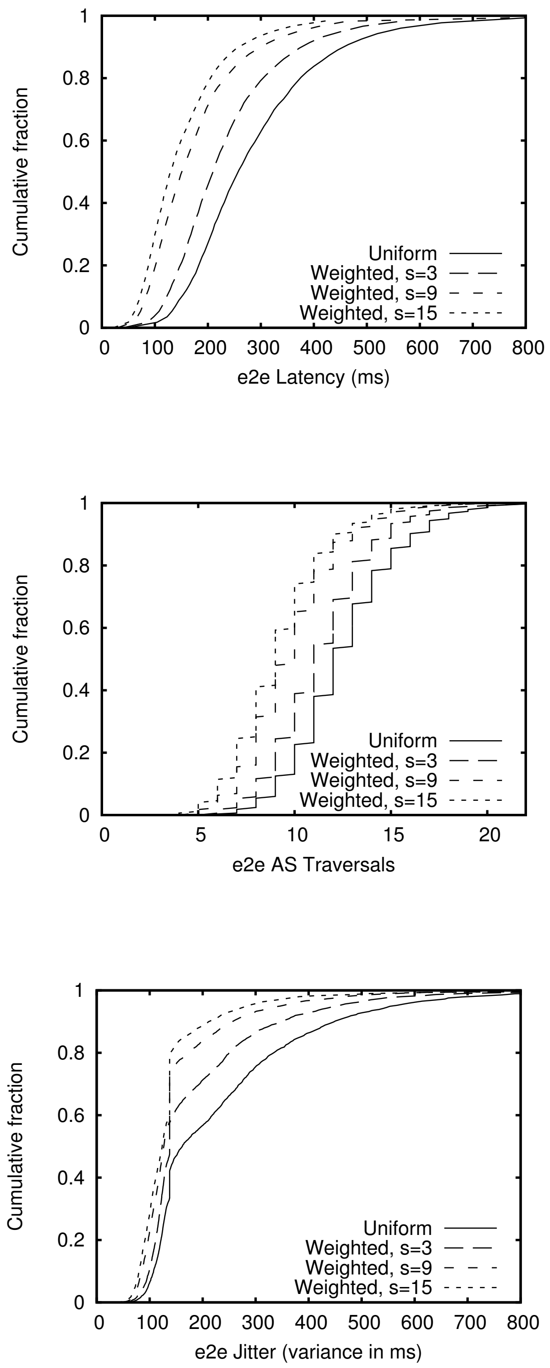


Figure 6.3: Path performance for the `King` *(top)*, `PL-ASes` *(middle)*, and `PL-Jitter` *(bottom)* datasets.

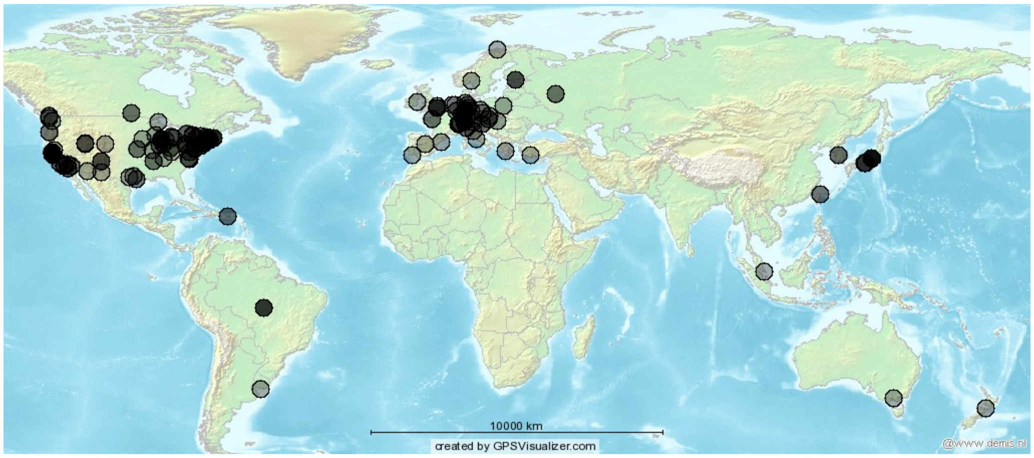


Figure 6.4: Locations of the 202 A^3 PlanetLab nodes. Locations are determined using the GeoIP [58] IP-to-geocoordinates library.

cache, irrespective of any performance implications. The WEIGHTED selection strategy offered improved performance (as indicated by a decrease in e2e latency). When $s = 15$ (recall that higher values of s bias the probability distribution more heavily in favor of performance), the median e2e RTT was 132ms as compared to 259ms for the UNIFORM strategy (a 49.0% reduction).

Similarly, A^3 is able to produce anonymous paths with fewer AS traversals. When run against our PlanetLab-based `PL-ASes` dataset, A^3 generated paths whose median number of AS crossings was 12 using UNIFORM and 9 when WEIGHTED was used and $s = 15$ (see Figure 6.3 *(middle)*).

Finally, as shown in Figure 6.3 *(bottom)*, A^3 significantly reduced the number of paths that exhibit large amounts of jitter (variance in ping interarrival times). For example, 43.2% of the paths generated by UNIFORM had jitter of 200 and higher. In contrast, the same jitter rates were experienced by only 28.8%, 16.1%, and 11.0% of paths outputted by WEIGHTED with $s = 3$, $s = 9$, and $s = 15$, respectively.

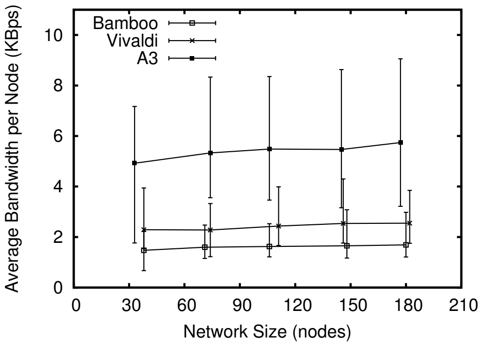


Figure 6.5: Bandwidth utilization on PlanetLab. Errorbars denote the 5th and 95th percentile ranges. The variance in network sizes between Bamboo, Vivaldi, and A^3 is due to the changing availability of PlanetLab nodes during experimentation.

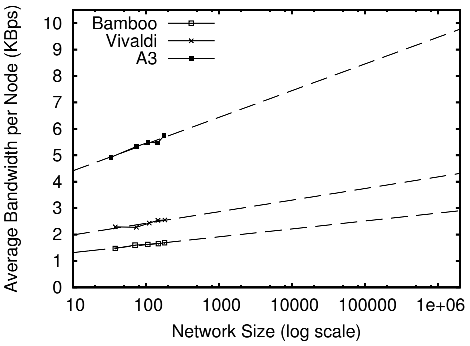


Figure 6.6: Predicted bandwidth utilization for large deployments, derived using logarithmic regression on measured values (see Figure 6.5). The corresponding regression equations are listed in Table 6.1.

6.4.3 PlanetLab Evaluation

In this section, we evaluate A^3's performance on real-world networks. We installed A^3 on 202 geographically disperse (see Figure 6.4) PlanetLab [74] nodes as well as three instances running in our university lab. For all experiments, all nodes joined within a three minute interval. Latency measurements were conducted at the application layer and include node processing time.

Bandwidth Overhead To maintain its virtual coordinate, each A^3 node periodically locates a neighbor using the distributed directory service, queries the located peer for its coordinate, and conducts an empirical measurement to that peer. In addition, A^3 nodes regularly fetch the coordinates of random peers using anonymous lookup tunnels to populate their anonymous router caches. In this section, we examine how much bandwidth is consumed by these functionalities.

Service	**Bandwidth equation** (KBps)	R^2
Bamboo (without Vivaldi or A^3)	$0.1299 \ln(N) + 1.017$	0.96
Bamboo with Vivaldi (without A^3)	$0.1909 \ln(N) + 1.545$	0.82
Bamboo with Vivaldi and A^3	$0.4392 \ln(N) + 3.402$	0.94

Table 6.1: Logarithmic regression analysis of PlanetLab bandwidth requirements. N denotes the number of peers in the network.

Figure 6.5 shows A^3's bandwidth utilization (measured as the average of all nodes' bandwidths) for different network sizes on PlanetLab. For comparison, the Figure also shows the bandwidth cost incurred by Bamboo running with neither Vivaldi nor A^3 ("Bamboo") as well as Bamboo running with Vivaldi but without A^3 ("Vivaldi"). Note that A^3's communication cost includes messages used to construct and deconstruct anonymous paths. (As described above, each relay's workload generator produces an anonymous path every five seconds.) Although A^3 users may request more paths in practice (consequently increasing A^3's bandwidth), our purpose here is to show A^3's bandwidth requirements during a fairly steady state, and to explore how these requirements vary for different network sizes. In our largest tested deployments, A^3 imposes a modest cost of only 3.2KBps over Vivaldi. In all cases, the bandwidth utilization is far less than the capacity of even dial-up Internet connections.

Our results also indicate that A^3 scales well with network size. When the number of nodes increased by a factor of more than 5 from 33 to 177, A^3's average bandwidth consumption increased by just 16% from 4.9 to 5.7KBps. This result is somewhat unsurprising, given that the distributed directory service routes `deliver` requests using $O(\log N)$ messages, where N is the network size.

Both the routing engine and the coordinate embedding system (the only two A^3 components that send periodic messages) utilize the `deliver` messaging function. Hence, the overall communication cost of using A^3 is also $O(\log N)$. The bandwidth

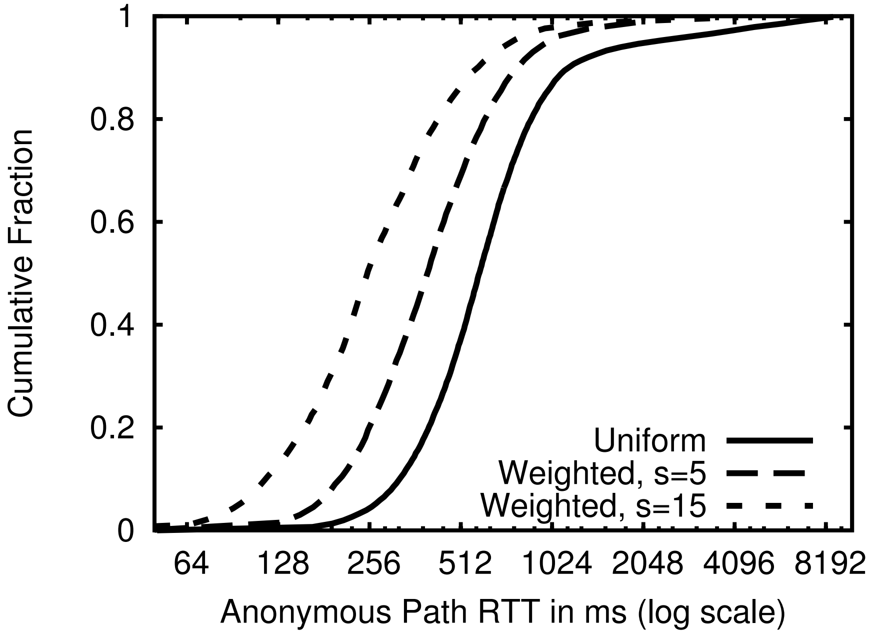


Figure 6.7: E2e path performance on PlanetLab.

requirements for much larger deployments can therefore be determined using logarithmic regression analysis. Figure 6.6 plots the average bandwidth per node for networks of up to one million peers. The empirical bandwidth measurements from Figure 6.5 are overlayed on the Figure. According to our regression analysis, the bandwidth requirement of an A^3 peer in a network with one million nodes is less than 10 KBps – far within the capabilities of broadband subscribers. Table 6.1 lists the bandwidth regression equations and their corresponding error values (R^2).

Path Performance Figure 6.7 shows the cumulative distribution of e2e path RTTs for all generated paths over a two-hour period. We tested A^3 using three configurations: Uniform relay selection, Weighted selection with $s = 5$, and Weighted selection with $s = 15$. As can be seen from the Figure, the link-based

WEIGHTED strategy produced anonymous paths with significantly lower RTTs than those produced using random selection. In particular, the median path RTTs were 591ms for UNIFORM, 403ms for WEIGHTED with $s = 5$ (a 31.8% reduction), and 253ms for WEIGHTED with $s = 15$ (a 57.2% reduction). Additionally, WEIGHTED decreased the percentage of high latency paths. For example, 64.8% of paths produced by UNIFORM had RTTs that exceeded 500ms. In comparison, only 32.5% of the routes generated by WEIGHTED with $s = 5$ had such high RTTs.

6.5 Summary

This chapter presented A^3, a flexible, secure and low-overhead architecture and platform for deploying anonymity-based services on the Internet. A^3 uses the link-based relay selection strategies introduced in Chapter 3 to produce paths with low latency and jitter and few AS crossings. Unlike existing anonymity systems such as Tor, the A^3 platform is fully decentralized and does not depend on *a priori* trusted nodes. Simulation results as well as experimentation on PlanetLab reveal that A^3 is effective at producing high performance anonymous paths. For example, in comparison to random relay selection, A^3's link-based strategies halve the RTTs of anonymous paths on PlanetLab. Furthermore, A^3 is highly scalable, requiring less than 10KBps of bandwidth per peer for a million-node deployment.

Chapter 7

Securing Coordinate Systems with Veracity

In Chapter 3, we introduced link-based relay selection techniques that intelligently select anonymous relays to produce high performance anonymous paths. To generate such routes, our WEIGHTED path selection algorithm estimates the e2e cost of candidate paths by aggregating the costs of each hop. Since there are $O(N^2)$ potential links in a network of N relays, maintaining pairwise distance information is untenable, particularly given that networks are not static and distances may (and do) change. To achieve practical link-based selection, we demonstrated in Chapter 4 that virtual coordinate embedding systems may be leveraged to linearize the information that must be shared among relays, enabling nodes to estimate pairwise distances given only N virtual coordinates. The A^3 anonymity system, introduced in Chapter 6, utilizes such an approach.

Since initiators use virtual coordinates to estimate path costs, the performance of link-based routing depends on the accuracy of the coordinate embedding system. Unfortunately, the distributed nature of coordinate systems make them particularly vulnerable to insider manipulation. For example, when 10% of the network is malicious, estimation errors for an unprotected Vivaldi coordinate system increase by a factor of nearly five [96]. When 30% of nodes misbehave, estimation errors increase by more than 1100% [96]. A corrupted coordinate system will, at best, decrease the performance of link-based routing since coordinate-based distance estimations will be inaccurate. At worst, a carefully orchestrated attack against the coordinate system may cause malicious nodes to appear more favorable, attracting an undue proportion of traffic.

This chapter presents *Veracity* [96], a *fully* decentralized service for securing network coordinate systems. Veracity provides a practical deployment path while providing equivalent (or greater) security than previously proposed coordinate security systems. Unlike prior proposals, Veracity does not require either pre-selected trusted nodes [44], the triangle inequality test [16], or outlier detection based on a fixed neighbor set [118], allowing Veracity to be practically deployed and react more rapidly to changes in network conditions. Veracity is also agnostic to the type of decentralized coordinate system being deployed, and can be employed as a protection service over existing decentralized coordinate systems [19, 16, 54, 66].

Veracity's fully decentralized architecture makes it ideally suited for A^3. Although Veracity is motivated by the need to protect A^3 from insider manipulation, Veracity is also applicable to other (non-anonymity) applications that rely on virtual coordinate systems, including proximity-based routing [99], neighbor selection in overlays [21], network-aware overlays [73], and replica placement in content-distribution networks [20, 112]. In addition to causing significantly decreased accuracy and performance, corrupted coordinate systems may serve as stepping stones for attacks against the applications that rely on them. Attackers who control the coordinate system may advertise attractive (but false) coordinates for nodes under their control, increasing the likelihood that such hosts will be selected for routes, neighbors, or replicas. Such compromises enable myriad attacks against the overlying services. For example, malicious nodes may misdirect intercepted messages sent via overlay routing, return false data when serving as a replica in a content distribution network, or partition the keyspace in a distributed hash table. Veracity protects such applications by preventing malicious nodes from injecting inaccuracy into the underlying coordinate system.

At a high-level, Veracity utilizes a two-step verification process. The first step involves a majority vote-based scheme in which a published coordinate has to be independently verified by a deterministically assigned set of *verification nodes* before it is used by peers. An adversary who attempts to disrupt the network by publishing inconsistent coordinates will fail this verification step, and consequently its coordinates will be ignored. As an additional measure, a second verification step utilizes a set of randomly chosen peers to independently compute the estimation error due to a new coordinate, and reject the coordinate if the error is above a threshold. This second protection mechanism detects attacks in which malicious nodes delay responses to measurement probes. The combination of the two techniques ensures that Veracity can tolerate a high fraction of malicious nodes that concurrently report false coordinates and delay latency measurements.

In this chapter, we focus our implementation and evaluation on Vivaldi since it is widely used [7], has been the focus of recent work [44, 118] on securing coordinate systems, and is used as the underlying coordinate system by A^3. We demonstrate via execution in a simulated network environment using realistic network traces [114, 47] and a deployment on PlanetLab that Veracity mitigates attacks for moderate sizes of malicious nodes (up to 30% of the network), even when coalitions of attackers coordinate their attacks. We further show that Veracity is resistant to high levels of churn and incurs only a modest communication overhead.

7.1 Comparison to Other Coordinate Protection Systems

Kaafar *et al.* [45] first identified the vulnerability of coordinate systems, in which just 5% of the participating nodes can render the system unusable simply by either lying about its coordinates or delaying RTT probe replies. Subsequently, there have been several recent proposals targeted at securing coordinate systems.

PIC detects dishonest nodes by observing that falsified coordinates or delayed measurements likely induce triangle inequality violations (TIVs) [16]. To verify peers' coordinates and measurements, honest nodes use distances to trusted landmarks to detect TIVs. Using a generated transit-stub topology of 2000 nodes, PIC is able to tolerate attacks when up to 20% of the network was controlled by colluding adversaries [16]. However, more recent work has indicated that TIVs can potentially be common and persistent [55], reducing the practicality of PIC's protection scheme on real-world networks.

Kaafar *et al.* propose the use of trusted *surveyor nodes* to detect malicious behavior [44]. Surveyor nodes position themselves in the coordinate space using only other trusted surveyors. Nodes profile surveyors to model honest behavior, detecting falsified coordinates and measurements as behavior that differs from their constructed model. Kaafar *et al.* conclude that their approach is effective when 30% or less of the network is controlled by malicious and cooperating nodes [44]. Their technique requires 8% of the nodes to be *a priori* trusted surveyors [44] – a nontrivial fraction when the network consists of 100,000 or more nodes.

The RVivaldi system proposed by Saucez *et al.* protects coordinate systems using surveyors as well as centralized *Reputation Computation Agents* (RCAs), the latter of which assigns reputations (trust profiles) to coordinates [89, 88]. Their technique is evaluated only against non-cooperating adversaries, and tolerates up to 20% malicious nodes [89].

Like Veracity, the system proposed by Zage and Nita-Rotaru is fully distributed and designed for potentially wide-scale deployments [118]. Their approach relies on outlier detection, reducing the influence of nodes whose coordinates are too distant (*spatial locality*) or whose values change too rapidly in short periods of time (*temporal locality*). Their technique successfully mitigates attacks when 30% or fewer of the nodes are under an attacker's control [118]. However, the temporal locality heuristic requires that each node maintains an immutable *neighborset*, a list of neighbors that a node uses to update its coordinates. Wide-scale deployments involving hundreds of thousands of nodes are likely to be dynamic with nodes frequently joining and leaving the system. The high rate of churn will lessen the opportunities for temporal analysis as nodes leave the system (since less history is available), and cause errors in such analysis for newly joined nodes for which frequent changes in coordinates are expected. In contrast, Veracity does not discriminate against spatial or temporal outliers, and as described in Section 7.7.2, tolerates high levels of churn.

7.2 Attacker Model

Prior studies [45, 44] have demonstrated that coordinate systems are susceptible to three classes of attacks: *disorder* attacks in which malicious insiders attempt to decrease the accuracy of the system by advertising false coordinates and delaying RTT responses, and *isolation* and *repulsion* attacks in which the attacker respectively attempts to isolate or repulse a subset of targeted nodes. Veracity's general approach defends against malicious nodes that falsify their coordinates or induce/report artificially inflated latencies. Hence, the techniques described in this chapter can mitigate all three attacks.

We adopt the constrained-collusion Byzantine model proposed by Castro *et al.* [12] in which malicious nodes can insert, delete, or delay messages. Given a network of size N, and some fraction ($f < 1$) of malicious attackers, there exist independent coalitions of size cN, where $1/N \leq c \leq f$.

7.3 Metrics

To assess the accuracy of a virtual coordinate, we utilize the metrics introduced in Section 4.1. The *median error ratio* of a node n_i is defined as the median over the *error ratios*

$$\frac{\left|D(n_i, n_j) - ||C_{n_i} - C_{n_j}||\right|}{D(n_i, n_j)}$$

between n_i and all other nodes n_j $(n_i \neq n_j)$, where $||C_{n_i} - C_{n_j}||$ denotes the Euclidean distance between the coordinates belonging to nodes n_i and n_j, and $D(n_i, n_j)$ is the measured distance (e.g., latency) between them. To quantitatively compare the performance and security of Veracity and Vivaldi, we define the *system error ratio* as the median over all peers' median error ratios. Finally, to show lower performance bounds, we also consider the *90th percentile error ratio* – i.e., the 90th percentile of nodes' median error ratios.

7.4 Overview of Veracity

We first provide an overview of Veracity's security mechanisms. We base our description on Vivaldi's coordinate update model. While other decentralized coordinate systems [16, 54, 66] differ in their implementations, the update models are conceptually similar to Vivaldi's, and hence, Veracity's techniques are applicable to these systems as well.

To update its coordinate, a participating node (the *investigator*) periodically obtains the coordinate of a selected peer (the *publisher*) and measures the RTT between the two nodes. In most implementations, the publisher is typically a preassigned neighbor node of the investigator. In Veracity, we relax the requirement that a publisher has to be a fixed neighbor of the investigator and instead use a distributed directory service (see Section 7.5.2) to enable investigators to scalably select random publishers on demand.

The basic update model of Vivaldi leads to two possible avenues of attacks:

First, if the publisher is dishonest, it may report inaccurate coordinates. Second, the publisher may delay the RTT probe response to increase the error of the investigator's updated coordinate. To defend against such attacks, Veracity protects the underlying coordinate system through a two-step verification process in which groups of nodes independently verify the correctness of another node's coordinates. We outline these two processes below, with additional details presented in Section 7.5.

- **Publisher coordinate verification:** When an investigator requests a coordinate from a publisher, a deterministic set of peers called the *verification set* (*VSet*) verifies the publisher's claimed coordinate. Veracity assigns each publisher a unique VSet. Each VSet member independently assesses the accuracy of the coordinate by conducting its own empirical measurements to the publisher and computes the coordinate's estimation error. If a majority of the VSet does not accept the publisher's coordinate, the investigator discards the coordinate.

- **Candidate coordinate verification:** Once an investigator verifies the publisher's coordinate, it proceeds to update its own coordinate based on its empirical RTT measurement between itself and the publisher. To detect cases in which the publisher purposefully delays the RTT probe response, the investigator updates its coordinate to a new one only if the new coordinate results in no more than a small increase in estimation error computed amongst an independent and *randomly chosen* set of peers (the *RSet*).

An important benefit of Veracity is that it makes no distinction between intentionally falsified coordinates and those that are inaccurate due to limitations of the coordinate embedding process. In either case, Veracity prevents the use of inaccurate coordinates.

7.5 Veracity Verification Protocols

This section presents details of Veracity's two-step verification protocol. We first focus on various mechanisms necessary to realize *publisher coordinate verification* that prevents investigators from considering inaccurate coordinates. We then motivate and describe the *candidate coordinate verification* that protects against malicious RTT probe delays by the publisher.

7.5.1 VSet Construction

When a Veracity node joins the network, it computes a *globally unique identifier* (GUID) by applying a collision-resistant cryptographic hash function H (e.g., SHA-1) to its network address. (To prevent malicious peers from strategically positioning their GUIDs, Veracity restricts allowable port numbers to a small range.) Given a node with GUID g, the members of its VSet are the peers whose GUIDs are closest to $h_1, ..., h_\Gamma$, determined using the recurrence:

$$h_i = \begin{cases} \mathrm{H}(g) & \text{if } i = 1 \\ \mathrm{H}(h_{i-1}) & \text{if } i > 1 \end{cases} \tag{7.1}$$

where i ranges from 1 to the *VSet size*, Γ. A larger Γ increases the trustworthiness of coordinates (since more nodes are required in the verification process) at the expense of additional communication.

VSet construction utilizes a hash function to increase the difficulty of stacking VSets with collaborating malicious nodes. Attackers who control large coalitions of peers may be able to populate a majority of a particular malicious node's VSet (for example, by strategically choosing IP addresses within its assigned range), but such VSet stacking requires at a minimum $\lceil \frac{\Gamma}{2} \rceil$ peers per VSet. In practice, many more malicious peers are required since the attacker does not have complete discretion over the IP addresses of its coalition members. Moreover, as nodes join and leave

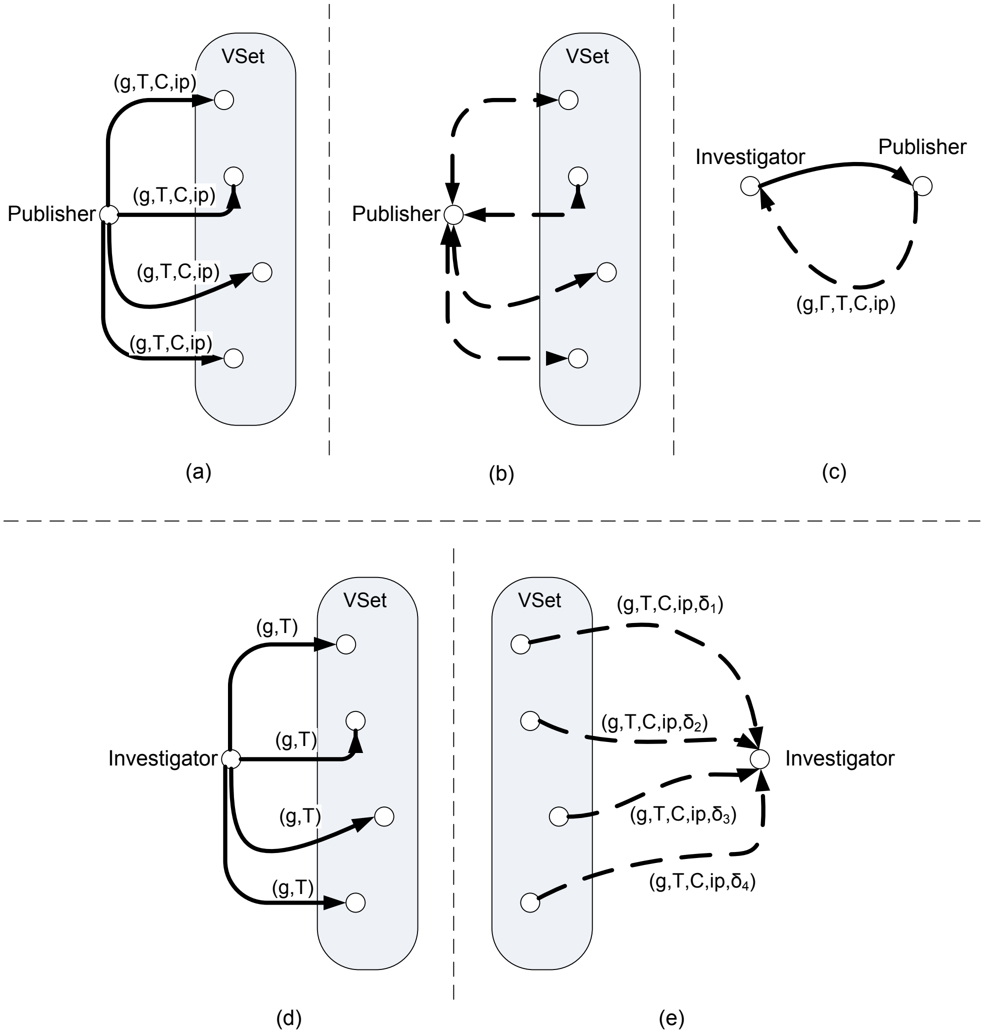


Figure 7.1: Publisher coordinate verification. Solid lines denote messages sent via `deliver` and dotted lines represent messages sent via direct IP. *(a)* Publisher distributes update tuple to VSet members using `deliver` messages addressed to GUIDs based on recursive hashes. *(b)* VSet members measure the RTT between themselves and Publisher. *(c)* Investigator queries Publisher and Publisher responds with claim tuple. *(d)* Investigator sends evidence query to Publisher's VSet members. *(e)* VSet members send evidence tuples to investigator.

the network, VSet members change (since the nodes whose GUIDs are closest to $h_1, ..., h_\Gamma$ also change), significantly impairing the ability to persistently stack VSets.

7.5.2 Locating and Updating VSet Members

Veracity utilizes a *distributed directory service* to resolve VSet members and route messages based on node GUIDs. Identical to the service described in Section 6.3.1 for A^3, the Veracity directory service implements a single API function, `deliver(`g, m`)`, which delivers the message m to the peer whose GUID is closest to g according to a keyspace distance metric. As with A^3, Veracity is compatible with any distributed directory service that supports the `deliver` function.

As shown in Figure 7.1(a), when a publisher updates its coordinate, it transmits an *update tuple* (g, τ, C, ip) to members of its VSet using the `deliver` function provided by the directory service. The update tuple contains the following values: g is the publisher's GUID, τ is a logical timestamp incremented whenever the publisher updates his coordinate, C is the new coordinate, and ip is the publisher's network address. Upon receiving the update tuple, each VSet member v_i measures the RTT between itself and ip (Figure 7.1(b)), and computes the *error ratio*

$$\delta_{(v_i,g)} = \frac{\left| RTT(v_i, ip) - ||C - C_{v_i}|| \right|}{RTT(v_i, ip)} \tag{7.2}$$

where C_{v_i} is v_i's coordinate and $||C - C_{v_i}||$ is the distance between the coordinates. Finally, v_i locally stores the *evidence tuple* $(g, \tau, C, ip, \delta_{(v_i,g)})$. Nodes periodically purge tuples that have not recently been queried to reduce storage costs.

7.5.3 Publisher Coordinate Verification

To update its coordinate, the investigator queries a random node (via a `deliver` message to a random GUID g) in the network (i.e., the publisher). As depicted in Figure 7.1(c), the publisher replies with a *claim tuple* (g, Γ, τ, C, ip). The investigator immediately discards the publisher's coordinates if the publisher's IP address is not ip, $g \neq \mathrm{H}(ip)$, or it deems Γ (VSet size) insufficiently large to offer enough supporting evidence for the coordinate.

Otherwise, the investigator transmits the *evidence query* (g, τ) to each member of the publisher's VSet, constructed on demand given g according to Eq. 7.1 (Figure 7.1(d)). If a VSet member v_i stores an evidence tuple containing both g and τ (logical timestamp), it returns that tuple to the investigator (Figure 7.1(e)). The investigator then checks that the GUID, network address, and coordinates in the publisher's claim tuple matches those in the evidence tuple. If there is a discrepancy, the evidence tuple is ignored.

After querying all members of the publisher's VSet, the investigator counts the number of non-discarded evidence tuples for which $\delta_{(v_i,g)} \leq \hat{\delta}$, where $\hat{\delta}$ is the investigator's chosen *ratio cutoff parameter*. Intuitively, this parameter gauges the investigator's tolerance of coordinate errors: a large $\hat{\delta}$ permits fast convergence times when all nodes are honest, but risks increased likelihood of accepting false coordinates. If the count of passing evidence tuples meets or exceeds the investigator's *evidence cutoff parameter*, R, the coordinate is considered verified. Otherwise, the publisher's coordinate is discarded.

7.5.4 Tuning VSet Parameters

To determine an appropriate value for the *ratio cutoff parameter* $\hat{\delta}$, we examined Vivaldi's system error ratio when run against the `Meridian` [114], `King` [47], and `S`^{3}`-Latency` [117] datasets, as well as a pairwise latency experiment (`PL-Latency`) that we executed on PlanetLab [74]. Table 7.1 provides the properties of the four datasets as well as Vivaldi's achieved system error ratio.

An appropriate value for $\hat{\delta}$ should be sufficiently large to accommodate baseline errors. For example, in our Veracity implementation (see Section 7.7.1), we use a ratio cutoff parameter $\hat{\delta}$ of 0.4, well above the system error ratio for all datasets.

In the absence of network churn, the VSet membership of a publisher remains unchanged. With network churn, some of the VSet members may be modified as

	# of	Pairwise Latency		System
Dataset	**Nodes**	**Avg.**	**Median**	**Err. Ratio**
Meridian	500	71.3	55.0 ms	0.15
King	500	72.7	63.0 ms	0.09
S^3-Latency	359	85.8	67.9 ms	0.17
PL-Latency	124	316.4	134.0 ms	0.10

Table 7.1: Properties of the Meridian, King, and S^3-Latency, and PL-Latency pairwise latency datasets, and Vivaldi's system error ratios for each dataset.

the keyspace of the directory service is reassigned. New VSet members may not have stored any evidence tuples, but as long as R (*evidence cutoff parameter*) VSet members successfully verify the coordinate, the coordinate can be used. In our experiments, we note that even when R is 4 for a VSet size of 7, Veracity can tolerate moderate to high degrees of churn while ensuring convergence in the coordinate system.

7.5.5 Candidate Coordinate Verification

The *publisher coordinate verification* scheme described in Section 7.5.3 provides the investigator with evidence that a publisher's *coordinate* is accurate. This does not prevent a malicious publisher from deliberately delaying an investigator's RTT probe, thereby causing the investigator to update its own coordinate erroneously. (Recall that to update its coordinate, the investigator must measure the RTT between itself and the publisher after having obtained the publisher's coordinate.)

Once an investigator has validated the publisher's coordinate, the *candidate coordinate verification* scheme compares coordinate estimation errors among the investigator and a random subset of nodes (the *RSet*) using the investigator's current coordinate (C_I) and a new candidate coordinate (C'_I) calculated using the publisher's verified coordinate and the measured RTT.

The investigator queries for the coordinates of Λ RSet members by addressing deliver messages to random GUIDs (Figures 7.2(a) and 7.2(b)). As with Γ (VSet

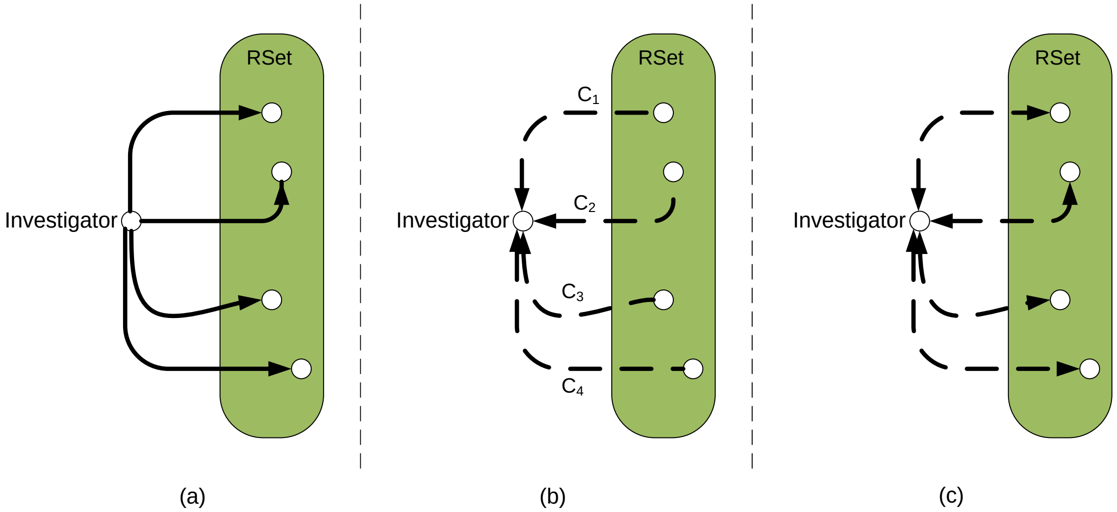


Figure 7.2: Candidate coordinate verification. Solid lines denote messages sent via `deliver` and dotted lines represent messages sent via direct IP. *(a)* Investigator queries random nodes (the RSet) for their coordinates. *(b)* RSet members report their coordinates to Investigator. *(c)* Investigator measures the RTTs between itself and RSet members, and then calculates the error ratios for the current (C_I) and candidate (C'_I) coordinates.

size), a larger Λ (RSet size) increases confidence in the candidate coordinate at the expense of additional communication. In our experimentation, we found that setting $\Lambda = \Gamma = 7$ provides reasonable security without incurring significant bandwidth overhead. The investigator (I) measures the RTT between itself and each RSet member (Figure 7.2(c)) and computes the average error ratio

$$err(C, \text{RSet}) = \frac{\left(\sum_{r_j \in \text{RSet}} \frac{\left|RTT_{Ir_j} - ||C - C_{r_j}||\right|}{RTT_{Ir_j}}\right)}{\Lambda} \tag{7.3}$$

for both C_I and C'_I. If the new coordinate causes the error ratio to increase by a factor of more than the *tolerable error factor* Δ, then C'_I is discarded and the investigator's coordinate remains C_I. Otherwise, the investigator sets C'_I as his new coordinate. The value of Δ must be sufficiently large to permit normal oscillations (e.g., caused by node churn) in the coordinate system. Setting $\Delta \geq 0.2$ enabled Veracity to converge at approximately the same rate as Vivaldi for all tested topologies (we investigate Veracity's effect on convergence time in Section 7.7.2).

7.6 Distributed Directory Services

Like A^3, Veracity utilizes DHTs to implement the `deliver` messaging functionality described in Section 7.5.1. While one can adopt a centralized or semi-centralized directory service, a fully-decentralized solution ensures scalability, allowing Veracity's security mechanisms to be deployable at Internet scale.

While DHTs ensure scalability, they are vulnerable to insider manipulation [110] due to their distributed nature. Dishonest nodes can attempt to increase their influence in the network by conducting Sybil attacks, or they can inject inaccurate query results using eclipse or routing attacks. Section 6.3.1 described a number of previously proposed DHT security schemes, all of which are compatible with Veracity's fully distributed design. Unforeseen attacks that manage to circumvent such mechanisms have the effect of artificially increasing the fraction of malicious nodes in the network (since a greater fraction of messages will be misdirected towards misbehaving nodes), and such attacks can be compensated for by increasing R (the number of VSet members that must support a publisher's claimed coordinate for it to be accepted) and Λ (the RSet size).

7.7 Implementation and Evaluation

In this section, we evaluate Veracity's ability to mitigate various forms of attacks in the presence of network churn. We have implemented Veracity by modifying the Vivaldi implementation that is packaged with Bamboo [7].

7.7.1 Experimental Setup

Veracity uses Vivaldi as the underlying coordinate system with a 5-dimensional coordinate plane (the recommended configuration in the Bamboo source code [7]). Each node attempted to update its coordinate every 10 seconds. The size of VSets and RSets were both fixed at 7 ($\Gamma = \Lambda = 7$). We used a ratio cutoff parameter $\hat{\delta}$ of 0.4 and an evidence cutoff parameter R of 4. That is, at least 4 of the 7 VSet members had to report error ratios less than 0.4 for a coordinate to be verified. The maximum tolerable increase in error (Δ) for the candidate coordinate verification was set to 0.2.

Our experiments are carried out using Bamboo's simulation mode as well as on PlanetLab. In the simulation mode, we instantiated 500 nodes with pairwise latencies from the `Meridian` dataset. To distribute the burden of bootstrapping peers, a node joins the simulated network every second until all 500 nodes are present. Nodes join via an already joined peer selected uniformly at random.

In our PlanetLab experiments, the 100 participating nodes joined within 3 minutes of the first node. The selected PlanetLab nodes were chosen in a manner to maximize geographic diversity. The simulation and PlanetLab experiments share a common code base, with the exception of the simulator's virtualized network layer.

In Sections 7.7.2 through 7.7.4, we present our results in simulation mode in the absence and presence of attackers, followed by an evaluation on PlanetLab in Section 7.7.5. We focus our evaluation on comparing Vivaldi (with no protection scheme) and Veracity based on the accuracy of the coordinate system, convergence time, ability to handle churn, and communication overhead.

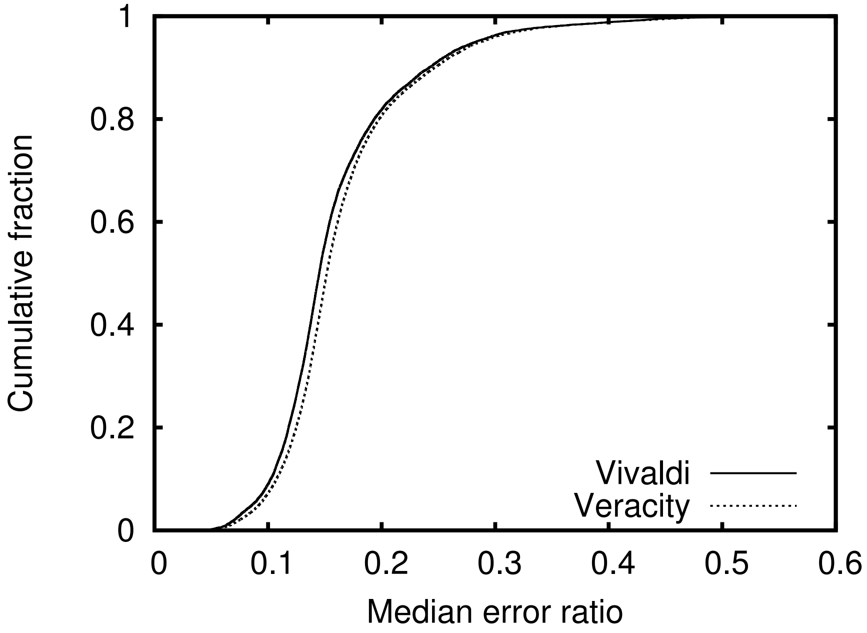


Figure 7.3: CDFs for median error ratios.

7.7.2 Veracity in the Absence of Attacks

Before evaluating the effectiveness of Veracity at mitigating various attacks, we first provide a performance comparison between Veracity and Vivaldi in the *absence* of any attackers within the simulation environment.

Accuracy of Network Coordinates Figure 7.3 shows the cumulative distribution functions (CDFs) of the median error ratios for Vivaldi and Veracity, computed after the system stabilizes. Veracity raises the system error ratio (the median of the nodes' median error ratio) by 4.6% (0.79ms) – a negligible difference given latencies over the wide-area. We observe that Veracity and Vivaldi have near identical CDFs, indicating that Veracity's protection schemes do not significantly influence nodes' coordinates in the absence of an attack.

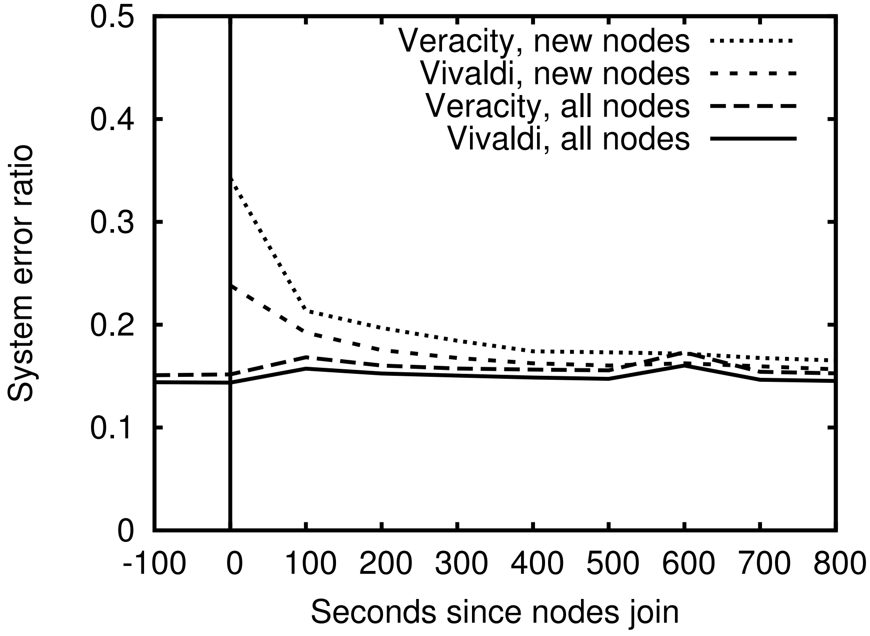


Figure 7.4: The system error ratio after 10 new nodes join the network (at $t = 0$). The median of the 10 new nodes' median error ratios is also shown. The coordinate system had stabilized prior to $t = -100$.

Convergence Time

To study how Veracity affects the rate at which the underlying coordinate system converges, we introduce 10 new nodes into the network after the remaining 490 peers have stabilized. Figure 7.4 plots the system error ratios for Vivaldi and Veracity before and after the new nodes join the network ("all nodes"). The system error ratios of both systems modestly increase when the new nodes are introduced and converge at approximately the same rate. The Figure also shows the median of the 10 new peers' median error ratios ("new nodes"). Although Veracity incurs a small initial lag in convergence time, the 10 new coordinates quickly reach within 15% of their stabilized (final) value in less than 200 seconds.

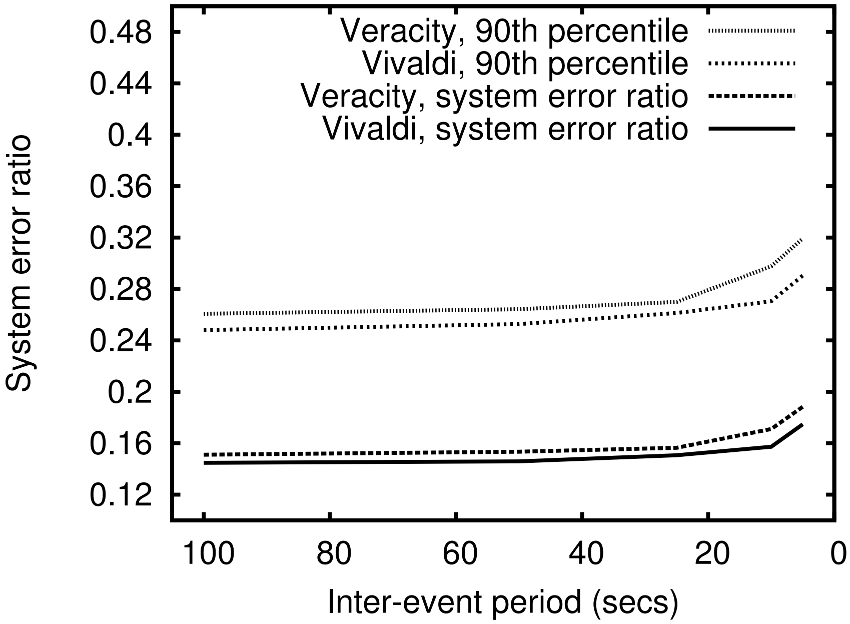


Figure 7.5: System error ratio for Vivaldi and Veracity under various degrees of churn.

The polling frequency – the rate at which nodes attempt to update their coordinate – is directly proportional to the system's convergence time. Higher polling frequencies enable faster convergence time at the expense of bandwidth. Although the values of the x-axis can be increased or decreased by adjusting the polling frequency, the shape of the curves remain fixed. Repeating our experiments with smaller and larger polling frequencies produced similar results.

Churn Effects

We next compare Vivaldi and Veracity's ability to handle churn. We adopt the methodology described by Rhea *et al.* [85] for generating churn workloads: a Poisson process schedules events ("node deaths") in which a node leaves the network. To keep the simulated network fixed at 500 nodes, a fresh node immediately takes the place of a node that leaves. The input to the Poisson process is the expected median inter-event period.

Figure 7.5 shows the system error ratio for Vivaldi and Veracity for various inter-event periods. Note that the level of churn is inversely proportional to the inter-event period. To illustrate near-worstcase performance, the figure also plots the 90th percentile error ratio.

Both Vivaldi and Veracity are able to tolerate high levels of churn. The "breaking" point of both systems occur when the inter-event period is less than five seconds, reflecting a rate at which approximately a quarter of the network is replaced every 10 minutes. Churn affects Veracity since the joining and leaving of nodes may cause the members of a VSet to more rapidly change, reducing the investigator's ability to verify a coordinate. Even at this high churn rate, Veracity's system error ratio (0.19) is only slightly worse than its error ratio (0.15) when there is no churn. It is worth emphasizing that such high churn (i.e., 25% of the network is replaced every 10 minutes) is unlikely for real-world deployments. The near 0-slope in Figure 7.5 for inter-event periods greater than 10 seconds shows that neither Vivaldi nor Veracity is significantly affected by more realistic churn rates.

7.7.3 Disorder Attacks

In this section, we evaluate Veracity's ability to mitigate *disorder attacks* in which malicious peers report a falsified coordinate chosen at random from a five dimensional hypersphere centered at the origin of the coordinate system. Points are chosen according to Muller's uniform hypersphere point generation technique [62] with distances from the origin chosen uniformly at random from $[0, 2000)$. Additionally, attackers delay RTT responses by between 0 and 2000 ms, choosing uniformly at random from that range. Malicious nodes immediately begin their attack upon joining the network.

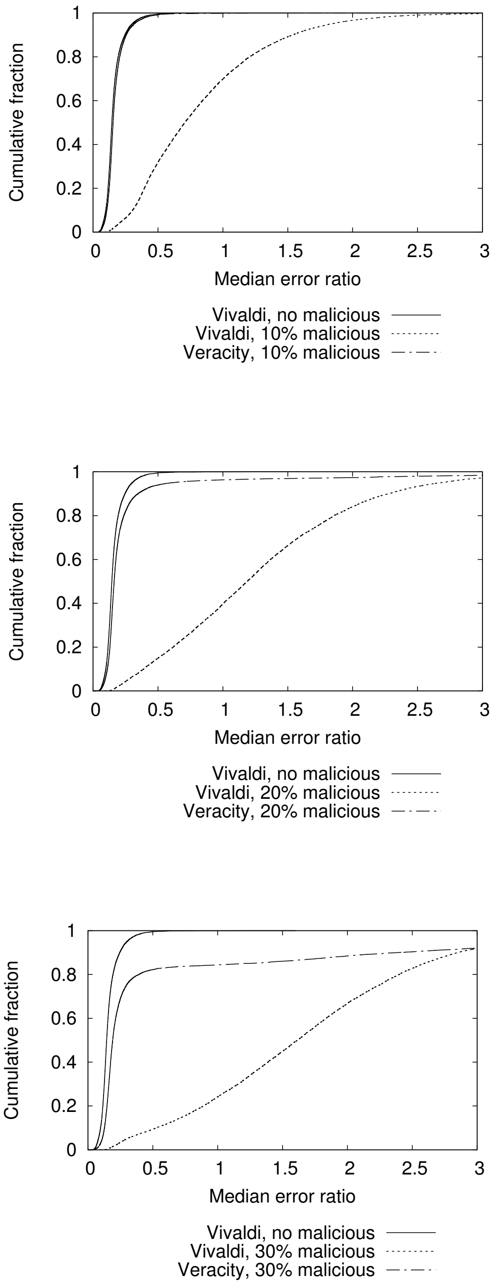


Figure 7.6: Honest peers' median error ratios achieved by Vivaldi and Veracity when malicious nodes constitute 10% *(top)*, 20% *(middle)*, and 30% *(bottom)* of the network. Median error ratios observed when using Vivaldi in a network with no attackers is shown for comparison.

To emulate realistic network conditions, all simulations experience moderate churn at a median rate of one churn event (a node leaving, immediately followed by a new node joining) every 120 seconds. This churn rate replaces 10% of the nodes during the lifetime of our experiments (100 minutes).

Uncoordinated Attacks

Figure 7.6 shows the effectiveness of Veracity at mitigating attacks when 10%, 20%, and 30% of peers are malicious. The attackers report a new randomly generated (and false) coordinate whenever probed, randomly delay RTT responses, and are *uncoordinated* (i.e., they do not cooperate). As our baseline, we also include the CDF for Vivaldi in the absence of any attackers.

Malicious attackers significantly reduce Vivaldi's accuracy, resulting in a 387% increase in the system error ratio (relative to Vivaldi when no attack takes place) even when just 10% of nodes are malicious. When 30% of nodes are malicious, the system error ratio increases dramatically by 1013%. In contrast, Veracity easily mitigates such attacks since the coordinate discrepancies are discernible in evidence tuples, causing inconsistently advertised coordinates to be immediately discarded by investigators. At low rates of attack (10%), the system error ratio increases by only 6% (representing a negligible system-wide median latency error of $4ms$). When 30% of the network is malicious, Veracity limits the increase in system error ratio to 32% ($5.7ms$), an 88% improvement over Vivaldi under the same attack.

Malicious nodes may conduct a more intelligent attack by randomly delaying probes while reporting *consistent* but erroneous coordinates. That is, each malicious node randomly generates a coordinate and reports the identical (and false) coordinate whenever probed. Such a strategy eliminates coordinate inconsistencies among VSet members. Compared to the previously described attack, this strategy results in lower estimation errors for Vivaldi but does slightly better against Veracity. Here, the increase in Vivaldi's system error ratio is 163% for 10% malicious nodes and 368%

for 30% malicious. Veracity successfully defends against heavy network infiltrations, yielding an increase in the system error ratio of just 39% when 30% of the network is malicious. Veracity reaches its tipping point when 40% of nodes are malicious, incurring an increase of 118%. We note that this increase is still far below the 497% increase experienced by Vivaldi.

Coordinated Attacks

We next consider *coordinated attacks* in which malicious nodes cooperate to increase the effectiveness of their attack. Malicious nodes offer supportive evidence for coordinates advertised by other dishonest nodes and do not offer any evidence for honest peers. That is, when queried, they provide evidence tuples with low (passing) error ratios for malicious nodes and do not respond to requests when the publisher is honest. We conservatively model an attack in which all malicious nodes belong to the same attack coalition. To further maximize their attack, each malicious node randomly generates a fixed erroneous coordinate and advertises it for the duration of the experiment. Additionally, attackers randomly delay RTT responses.

Figure 7.7 shows Veracity's performance (measured by the cumulative distribution of median error ratios) when the malicious nodes cooperate. For comparison, the Figure also plots the CDFs for equally sized uncoordinated attacks against Veracity and Vivaldi. Since Vivaldi does not collaborate with peers to asses the truthfulness of advertised coordinates, there is no equivalent "coordinated" attack against Vivaldi.

For all tested attack strengths, the coordinated attacks did not induce significantly more error than uncoordinated attacks. The resultant system error ratios differed little: when attackers control 30% of the network, the system error ratios are 0.202 and 0.201 for the uncoordinated and coordinated attacks, respectively (for comparison, Vivaldi's system error ratio is 0.679).

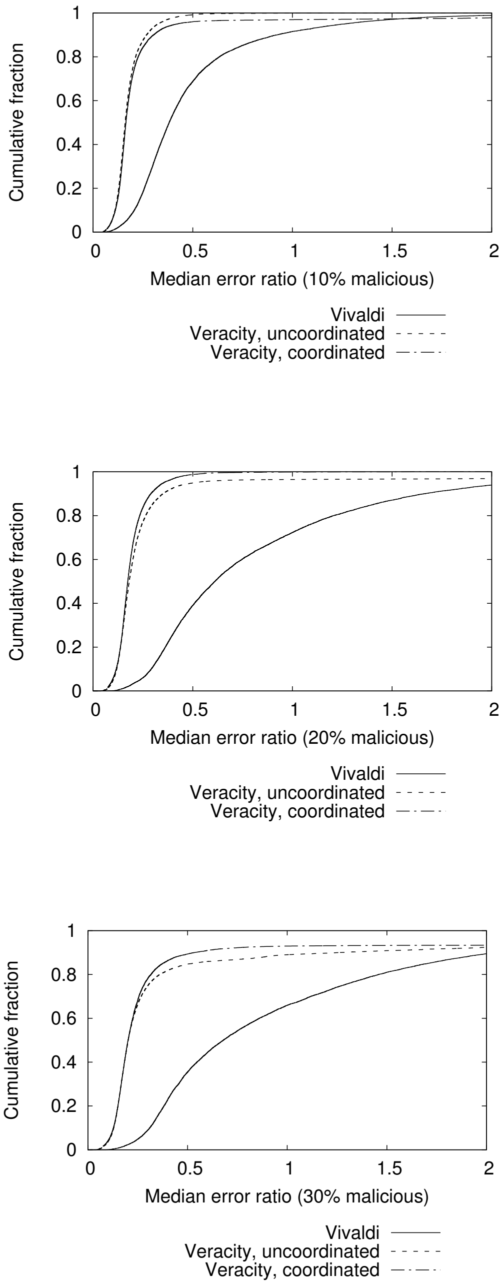


Figure 7.7: Honest peers' median error ratios when attackers conduct uncoordinated and coordinated attacks. Attackers comprise 10% *(top)*, 20% *(middle)*, and 30% *(bottom)* of network peers.

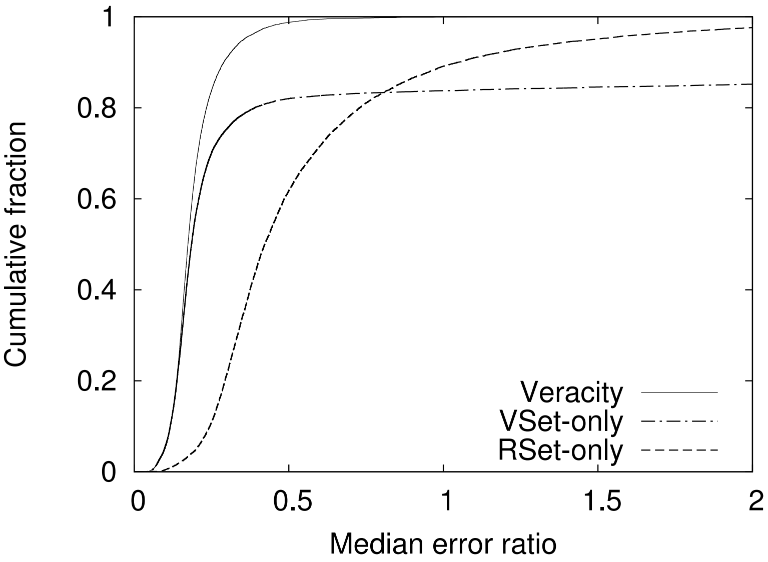


Figure 7.8: CDF of median error ratios for Veracity, Veracity without Candidate Coordinate Verification ("VSet-only"), and Veracity without Publisher Coordinate Verification ("RSet-only"). The attacker controls 20% of the network and conducts a coordinated attack.

Rejected: VSet-only and RSet-only Veracity

The previous sections show that Veracity's two protections schemes – publisher coordinate verification and candidate coordinate verification – effectively mitigate attacks when the adversary controls a large fraction of the network. In this section, we investigate whether it is sufficient to apply only one of the two techniques to achieve similar security.

Figure 7.8 shows the cumulative distribution of median error ratios when nodes utilize only publisher coordinate verification ("VSet-only") or candidate coordinate verification ("RSet-only"). We model the attack scenario from Section 7.7.3 in which 20% of the nodes are malicious and cooperating. For comparison, we also show the CDF when both strategies are utilized ("Veracity"). The VSet-only technique

Percentage of malicious nodes	Inconsistent coords (Uncoordinated)		Consistent coords (Uncoordinated)		Consistent coords (Coordinated)
	Vivaldi	Veracity	Vivaldi	Veracity	Veracity
0%	1.00	1.05	1.00	1.05	1.05
10%	4.87	1.06	2.63	1.11	1.10
20%	8.18	1.12	4.21	1.25	1.22
30%	11.13	1.32	4.68	1.39	1.48
40%	13.90	1.54	5.97	2.18	2.37

Table 7.2: Relative system error ratios (system error ratio of the tested system divided by the system error ratio of Vivaldi when no attack takes place) for various attacker scenarios.

achieves nearly the same system error ratio as Veracity (0.19 and 0.17, respectively). However, using only publisher coordinate verification results in a very long tail of median error ratios. In particular, the 90th percentile error ratio is 0.29 for Veracity and 4.42 for VSet-only. Hence, publisher coordinate verification protects the accuracy of most nodes, but permits a significant degradation in accuracy for a minority of peers.

By itself, candidate coordinate verification results in a higher system error ratio (0.42) than VSet-only or Veracity. Additionally, RSet-only has a longer tail than Veracity, resulting in a 90th percentile error ratio of 1.05 during the attack.

By combining both techniques, Veracity better protects the underlying coordinate system, achieving error ratios that nearly mirror those produced by Vivaldi in the absence of attack (see Figures 7.6 and 7.7).

Summary of Results

To summarize the performance of Veracity under disorder attacks, Table 7.2 shows the *relative system error ratio* for various attacker scenarios that we have described, where each system error ratio is normalized by that obtained by Vivaldi under no attacks.

Overall we observe that Veracity is effective at mitigating the effects of disorder

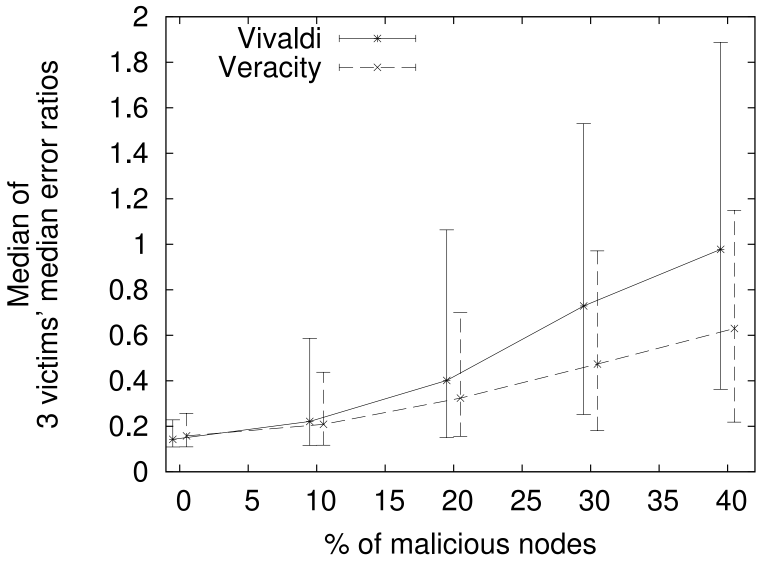


Figure 7.9: Effects of a combined repulsion and isolation attack against three victim nodes. Points represent the median with error bars denoting the 10th and 90th percentile of the median error ratios of the three victims. For readability, datapoints are slightly shifted along the x-axis by -0.5 for Vivaldi and $+0.5$ for Veracity.

attacks. Even under heavy attack (40% malicious nodes), disorder attacks result in a relative system error of 1.54, far below Vivaldi's relative median error of 13.9.

Veracity's effectiveness matches or exceeds that of the prior proposals discussed in Section 7.1. In contrast to existing coordinate protection systems, Veracity does not require pre-selected trusted nodes, triangle inequality testing, nor outlier detection based on a fixed neighbor set, and is therefore better suited for practical deployment.

7.7.4 Repulsion and Isolation Attacks

While Veracity is intended primarily to defend against disorder attacks, our next experiment demonstrates the effectiveness of Veracity for protecting against repulsion and isolation attacks. We carry out a combined repulsion and isolation attack as follows: malicious nodes are partitioned into three coalitions, each of which attempts to repulse and isolate a single victim node. Attackers attempt to repulse the targeted node towards an extremely negative coordinate (i.e., having −1000 in all five dimensions) by using the following heuristic: if the victim is closer than the attacker to the negative coordinate, the attacker behaves honestly. Otherwise, the attacker reports his accurate coordinate but delays the victim investigator's RTT probe response by 1000ms, causing the victim to migrate his coordinate (provided it passes candidate coordinate verification) towards the negative coordinate.

Figure 7.9 shows the median of the three victim nodes' median error ratios achieved during the combined repulsion and isolation attack. In contrast to previous experiments, we do not use the system error ratio (the median over all peers' median error ratios), as repulsion and isolation attacks target specific victims and need not cause a significant degradation in coordinate accuracy for the remaining peers.

Veracity consistently offers lower median error ratios than Vivaldi. While Veracity does not completely mitigate the effects of repulsion and isolation attacks, our results suggest that the vote-based verification scheme is amenable to defending against such attacks.

7.7.5 PlanetLab Results

In our last experiment, we validate our simulation results by deploying Veracity on the PlanetLab testbed.

Communication Overhead

To quantitatively measure Veracity's communication overhead in practice, we analyze packet traces recorded on approximately 100 PlanetLab nodes for both Vivaldi and Veracity. Traces are captured using tcpdump and analyzed using the tcpdstat network flow analyzer [30]. Figure 7.10 shows the per-node bandwidth (averaged

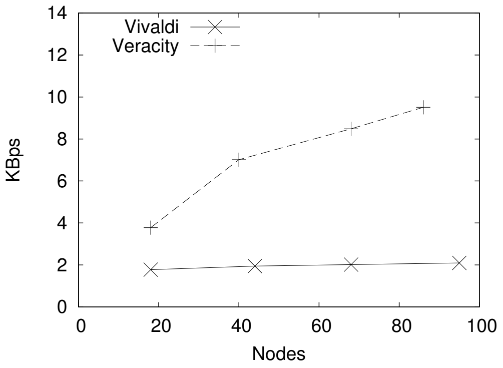


Figure 7.10: Bandwidth (KBps) on PlanetLab.

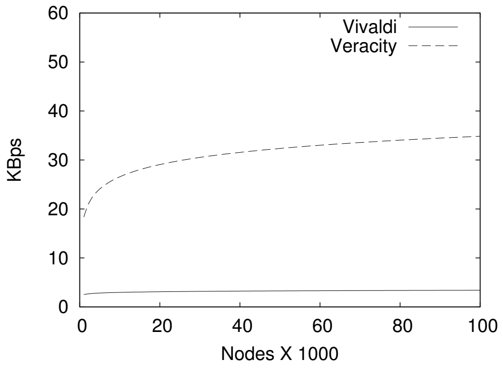


Figure 7.11: Extrapolated bandwidth (KBps) for large networks.

over all nodes) utilization (KBps) for Vivaldi and Veracity.

Veracity incurs a communication overhead since publishers' coordinates must be verified by VSets and investigators' candidate coordinates must be assessed by RSets. Since Veracity uses a DHT as its directory service, it leverages the scalability of DHTs: each verification step requires $O((\Gamma + \Lambda)\log_2 N)$, where Γ and Λ denotes the VSet and RSet sizes respectively.

Based on the PlanetLab measurements, we performed logarithmic regression analysis to extrapolate the per-node bandwidth requirements of Veracity as the number of nodes increases: $0.1895 \log N + 1.228$ KBps ($r^2 = 0.998$) for Vivaldi and $3.591 \log N - 6.499$ KBps ($r^2 = 0.994$) for Veracity. Figure 7.11 shows the extrapolated bandwidth utilization of Vivaldi and Veracity for large networks. For a large network consisting of 100,000 nodes, Veracity's expected per-node bandwidth requirement is a modest 35KBps, making it accessible to typical broadband users.

Accuracy Under Disorder Attacks

Figure 7.12 plots the system error ratio achieved on PlanetLab for varying attacker infiltrations. Malicious nodes advertise inaccurate (but consistent) coordinates, delay

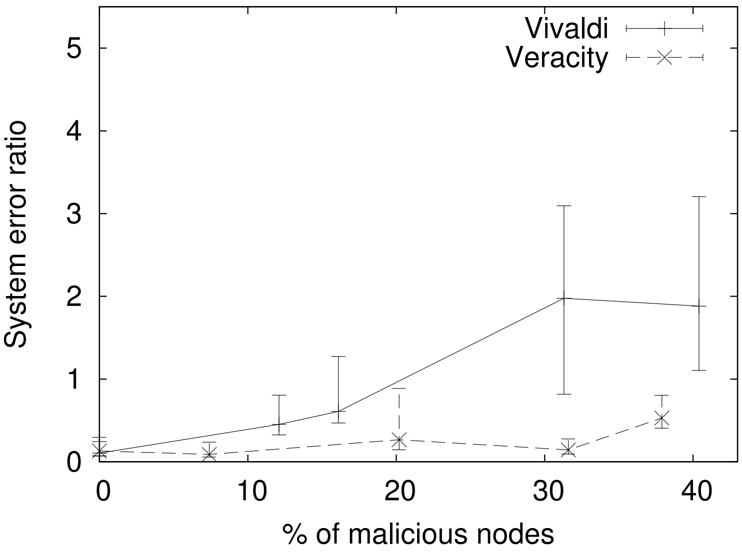


Figure 7.12: System error ratio achieved on PlanetLab. Error bars denote the 10th and 90th percentile error ratios.

RTT responses, and do not coordinate their attack. To calculate error ratios (which requires knowledge of actual pairwise RTT measurements), we conducted pairwise ping measurements on the PlanetLab testbed. We observe that Veracity effectively mitigates attacks, yielding an increase in system error ratio (relative to Vivaldi under no attack) of just 38% when 32% of the network is malicious. In contrast, Vivaldi suffers an increase of 1679% when 31% of the nodes are dishonest. (The slight differences between attacker percentages is due to the intermittent availability of PlanetLab nodes.)

7.8 Summary

This chapter proposes *Veracity*, a fully distributed service for securing network coordinates. We have demonstrated through extensive network simulations on real pairwise latency datasets as well as PlanetLab experiments that Veracity effectively mitigates various forms of attack. For instance, Veracity reduces Vivaldi's system error ratio by 88% when 30% of the network misbehaves by advertising inconsistent coordinates and adding artificial delay to RTT measurements. Veracity performs well even against cooperating attackers, reducing Vivaldi's system error ratio by 70% when 30% of the network is corrupt and coordinates its attacks.

Veracity provides a more practical path to deployment while providing equivalent (or greater) security than previously proposed coordinate security systems. Unlike PIC, Veracity does not associate triangle-inequality violations (TIVs) with malicious behavior [16], and as indicated by our simulation and PlanetLab results, does not impose additional inaccuracies in the coordinate system when TIVs do exist. Veracity is fully decentralized, requiring no *a priori* shared secrets or trusted nodes. In comparison to techniques that require specialized trusted nodes [89, 88, 44], Veracity is well-suited for fully distributed applications (in particular, A^3) in which centralized trust models are incompatible.

Chapter 8

Contour: Coordinate-Based Routing for Performance

The A^3 anonymity system uses coordinate-based distance estimations to produce high performance anonymous routes. In this chapter, we present *Contour*, a routing infrastructure that leverages virtual coordinate embedding systems to enable senders to construct non-anonymous paths whose end-to-end performance *exceeds* that of direct communication to the receiver.

8.1 Background: Detour Routing

Detour routing has been proposed as a means of increasing the reliability [4, 41] and performance [49, 90] of Internet communication. In detour routing, the sender relays her traffic through one or more relay nodes, forming routes either on a network overlay or through a specialized network routing infrastructure. Reliability is typically achieved by aggressively rerouting traffic along alternative paths [4, 41]. Detour routing has also been used for performance-based routing [49, 90] by taking advantage of the prevalence of triangle inequality violations (TIVs) on the Internet [53, 91, 120, 121] to identify alternative routes that lead to decreased latency or increased throughput.

There have been several proposals on using detour routing for reliability and performance. In Savage *et al.*'s seminal work on detour routing [90], intelligent detour routers are required at key access and interchange points. Traffic is routed to the nearest detour router which then forwards the traffic along tunnels within the detour network.

The Resilient Overlay Network (RON) [4] proposes building a detour network at the application-layer, requiring no changes to the underlying network infrastructure. However, RON requires pairwise measurements to locate alternative routes around failures. The number of such measurements grows geometrically with the size of the network, and hence limits the scale of deployment.

While the recently proposed *Scalable One-hop Source Routing* (SOSR) [41] has improved scalability, its utility is primarily focused on improving reliability. The lack of scalability is particularly significant since TIVs are more prevalent in large networks, hence providing increased opportunities from benefiting from seeking alternative paths.

Coordinate systems such as Vivaldi [19], PIC [16], ICS [54], and NPS [66] have been proposed in recent years, and their primarily usage has been for neighbor selection [21] and replica placement [20, 112], and only as a means to estimate pairwise latencies. Rakotoarivelo *et al.* suggests the benefits of using coordinate systems to improve QoS in a peer-to-peer network [78]. Lumezanu *et al.* propose the PeerWise [56] system for finding mutually advantageous one-hop detour paths. Like Contour, PeerWise uses virtual coordinate systems to identify TIVs and find candidate relays. To our best knowledge, Contour is the first scalable architecture that uses coordinate systems to improve the performance of Internet routing, with implementation and deployment on an actual testbed.

Dataset	Nodes	Avg. RTT	Pairwise links	% with TIV
Meridian	2500	151 ms	3122867	95.89%
King	1740	181 ms	1499979	85.56%
S^3-Latency	365	400 ms	109615	50.62%

Table 8.1: Triangle inequality violations (TIVs) for various latency datasets.

8.2 Motivation: Trace-driven Analysis

As has been well-documented in prior work [48, 69, 91, 56], triangle-inequality violations (TIVs) are commonplace on the Internet. We confirmed the prevalence of TIVs by finding pairs of nodes from the Meridian [114], King [47, 42], and S^3-Latency [117] latency datasets for which there is at least one indirect path with an e2e latency shorter than that of the corresponding direct route. (Unlike in previous chapters, we consider all nodes – not just the first 500 – in the three datasets.) Table 8.1 lists the percentage of pairwise connections in which there exists at least one TIV.

Our analysis further empirically quantifies the *maximum performance improvement* that one can reap from a one-hop relay. (Previous work has demonstrated that little performance may be gained through detour routes with more than one detour [56].) While this analysis serves as an upper bound, it allow us to conduct a feasibility study on how much one can gain from detours, and more importantly, provides a basis of comparison for the Contour system.

Let N represent the set of all nodes in the network and XY denote the latency between nodes $X, Y \in N$. Given a network-wide view consisting of all $|N \times N|$ distances, we can easily compute the optimum single-hop latency detour route for each node pair $(A, B) \in N \times N$ by finding the node $R \in N \setminus \{A, B\}$ that minimizes the latency $AR + RB$. The *maximum RTT improvement* is calculated as

$$2 \cdot \frac{(AB - (AR + RB))}{AB} \tag{8.1}$$

Note that Equation 8.1 assumes that network distances are symmetric (and hence

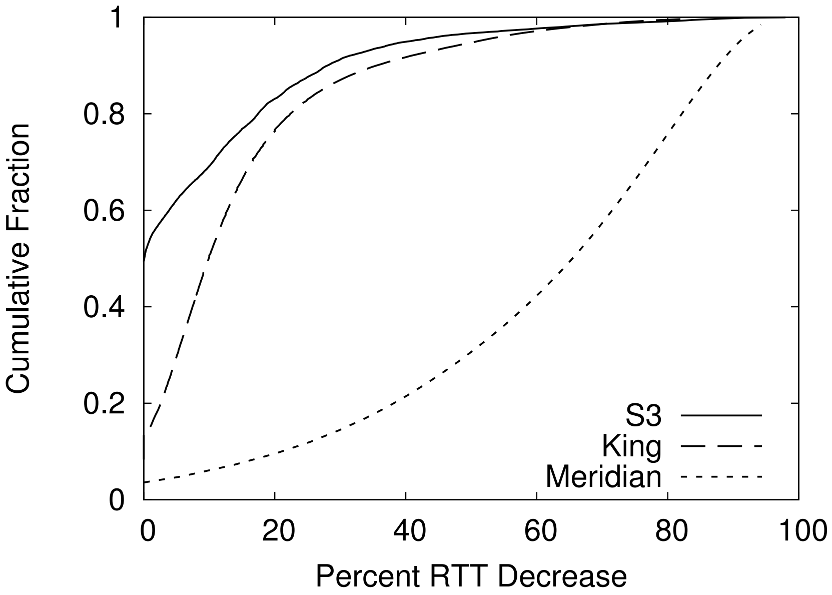


Figure 8.1: CDF of the percentage improvement (decrease) in RTT achieved by selecting the optimal single-hop detour relay.

the RTT of a link is twice its latency). Our assumption mirrors that of virtual coordinate systems, in which the network cost of routing from node A to node B is equal to the cost of routing from B to A.

Figure 8.1 shows the CDF of the maximum RTT improvements achieved via the optimal detour path. As expected, the `Meridian` dataset (in which 96% of links experience TIVs) exhibits the largest improvement. By comparison, the median maximum RTT improvement is only 0.09% for the $\texttt{S}^3$`-Latency` dataset, which is unsurprising given that only 50.6% of links exhibit a TIV. (As indicated in the last column of Table 8.1, larger networks contain higher percentages of links with TIVs. This is not unexpected given that larger networks provide more opportunities to find detour routes.)

The above optimality analysis serves as an upper bound for performance. In

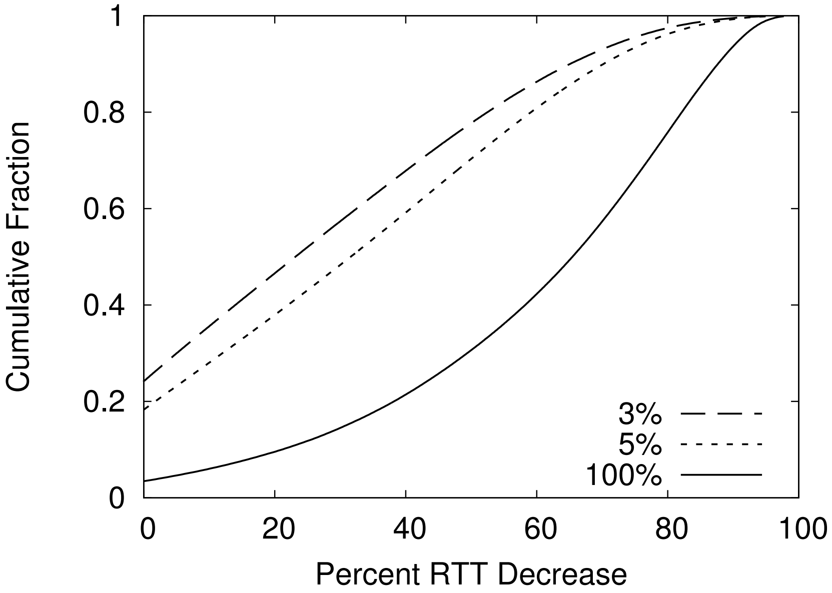


Figure 8.2: CDF of percentage RTT improvement (decrease) with knowledge of 3%, 5%, and 100% of all nodes in the `Meridian` dataset.

practice, selecting an optimal relay node in this fashion is not feasible, since global knowledge of the network is required. As network size increases, so do the opportunities to generate detour routes that produce significantly improved communication performance; however, since the number of pairwise distances grows geometrically with network size, sharing such information among all nodes is infeasible for large networks, particularly given that network changes occur frequently and must be propagated to all peers. We now consider whether it is feasible to select a *good* relay node given only a small subset of the network. This allows us to tradeoff route optimality and communication overheads in relay selection, and will be a basis for the coordinate-based selection strategies described in the next section.

Figure 8.2 shows the cumulative distribution of the percentage decrease in RTT achieved by detour routes when the sending node only knows the existence of 3%

(75 nodes), 5% (125 nodes), and 100% (2500 nodes) of the peers in the `Meridian` network. That is, the sender knows the RTTs between itself and a relay and between the relay and the receiver, but knows only a small fraction of such relays. Even when senders have knowledge of only 3% of potential detour relays, the median percent reduction in RTT is 23%, and 20% of direct paths could reduce their RTTs by 52% or more by routing through a known detour.

Although Figure 8.2 is somewhat optimistic as it assumes that nodes know exact distances between peers, it illustrates that detour routing can deliver significantly improved e2e performance even when senders have knowledge of only a small subset of nodes. Hence, rather than attempt to provide nodes with a complete network graph, Contour enables nodes to produce high performance detour paths using only a small subset of information about the network.

8.3 System Architecture

In this section, we describe the Contour architecture, followed by a description of relay selection strategies.

8.3.1 Overview

Figure 8.3 shows an overview of a Contour deployment. In our figure, a sender (Alice) wishes to send a message to a receiver (Bob). Alice actively participates in Contour, and thus may generate detour paths that attempt to meet her specific, per-connection e2e communication performance criteria (e.g., latency, loss, etc.). To send a message to Bob, Contour selects k detour relays, each of which is used to form independent single-hop *detour paths* to Bob. In the absence of an improvement via the use of a detour, the default Internet route (the *direct path*) will be used.

Contour may be configured as to the level of message duplication among the

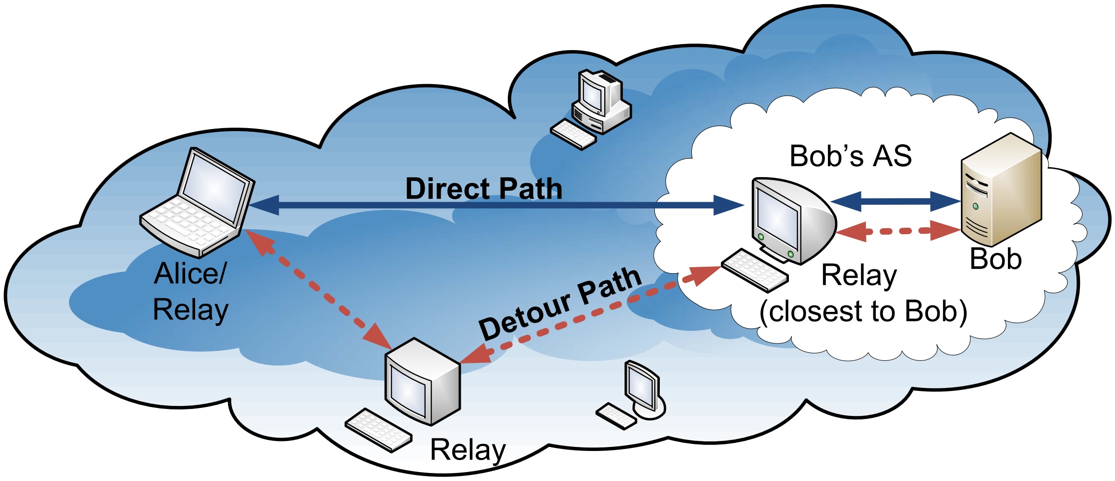


Figure 8.3: Contour architecture.

$k+1$ paths. To maximize performance and resiliency, messages may be duplicated and sent via each path, with Bob processing the first arriving copy of each message. Of course, such a technique imposes a substantial bandwidth overhead, and only achieves maximal performance if Alice, Bob, and the intermediate routers between them have the available bandwidth to send and receive $k+1$ streams. Alternatively, Contour may measure the e2e performance characteristics of each of the $k+1$ paths and send messages through the best performing subset of paths. The number of utilized paths is an application-tunable parameter. Measurements may be periodically repeated to ensure that the best performing paths are used.

Contour does not require Bob's active participation (consequently, Alice can utilize Contour to produce high performance paths to arbitrary networked services). However, since he does not actively participate, Bob cannot be the endpoint of the direct and detour paths. If Bob is not a relay node, the Contour node closest to him functions as a transparent proxy, forwarding messages to Bob and sending his responses to the upstream relay. For such cases, Alice must direct her detour paths to Bob's closest node. There are several well-studied techniques to locate a node close to a particular target. For example, OASIS maps IP prefixes to geographic coordinates of known landmarks [38], and the ClosestNode [115] service uses Meridian [114] to locate the closest server to the requesting client. We consider the challenge of locating the closest relay node to be orthogonal to the design of Contour.

To scalably generate detour paths, Contour first requires a mechanism to uniquely identify all nodes, and to select a relay node from the network. Each Contour node (also called a *relay*) registers with a *distributed directory server*. As with A^3 and Veracity, Contour requires that the distributed directory server implement the `deliver`(g, m) function that delivers message m to the node whose globally unique identifier (GUID) is closest to g. Also like A^3, Contour GUIDs are calculated by taking a cryptographic hash (e.g., SHA-1) of nodes' network addresses, and should therefore be roughly uniformly distributed among some large space. Although Contour is compatible with any consistent hashing service, our implementation uses *distributed hash tables* (DHTs) [6] due to their scalability, geographic distribution, load-balancing properties, and resilience to network churn [85].

Rather than locate potential relays on demand, Contour periodically searches the distributed directory service for random nodes, storing the resolved network addresses in a local cache. To preserve freshness, old cache entries are periodically purged. Shorter timeouts ensure cache entries are more fresh, but reduce the number of peers that may serve as relays.

To produce a high performance path to Bob, Contour selects k entries from its cache according to a *relay selection strategy* (described below) and forms k independent single-hop detour paths to Bob using each selected node as the intermediary relay of a path. Since detour paths do not always provide improved performance, Alice also creates a direct path to Bob.

8.3.2 Relay Selection Strategies

Below, we introduce the Rand and Coord relay selection strategies.

Random Relay Selection Strategy (Rand) Rand is a simple strategy in which Contour selects k nodes uniformly at random from its cache. Despite its simplicity, we note that Rand is typically sufficient for cases in which the absolute increase in performance is not critical (e.g. routing around failures), or in cases where TIVs are so prevalent that picking the best among k nodes suffices.

Coordinate-based Relay Selection Strategy (Coord) The Coord strategy utilizes virtual coordinate systems (e.g., Vivaldi [19]) to select candidate relay nodes. As described above, Contour periodically uses the distributed directory service to locate random peers and populate its cache. For each metric (available bandwidth, latency, etc.), Alice searches her cache for the k entries $X_1^{\text{metric}}, ..., X_k^{\text{metric}}$ that yield the k smallest values of

$$||C_{\text{A}} - C_{X_i^{\text{metric}}}|| \odot ||C_{X_i^{\text{metric}}} - C_{\text{B}}|| \tag{8.2}$$

where C_{A} is Alice's coordinate, $C_{X_i^{\text{metric}}}$ is the coordinate of cache entry X_i^{metric}, and C_{B} is the coordinate of Bob (if he participates) or the Contour node closest to Bob (if he does not). C_B is obtained by polling the relevant node. The $\odot$ function represents a *path concatenation operation* and is dependent of the particular metric being used. Table 3.1 lists formulas for computing the cost of a path with h hops with corresponding distances $d_1, ..., d_h$. For example, $\odot$ is *summation* for RTT and *minimum* for bandwidth.

To rank relays using multiple network metrics, Alice assigns a weight $w_i \geq 0$ to each metric $m_1, ..., m_z$. If $c_1, c_2, ..., c_z$ represents the estimated costs of using a cache entry as a detour node for metrics $m_1, ..., m_z$, she calculates the weighted cost of using that entry to be $\sum_{i=1}^{z}(w_i \cdot c_i)$. Cache entries are sorted based on their weighted costs, with the entries with the k smallest costs being chosen as the detour nodes.

8.4 Evaluation

To evaluate Contour's ability to produce high performance detour paths, we implemented Contour using a modified version of the Bamboo DHT [7].

Vivaldi is configured to use a five dimension coordinate space, the recommended configuration specified in the Bamboo source code [7]. Contour nodes locate a random peer and cache the peer's coordinate every five seconds. Cached entries persist in the cache for fifteen minutes before being purged. All participating nodes serve as potential relay nodes. As a workload generator, each node generates a single-hop detour route every five seconds.

8.4.1 Trace-driven Simulation

The first set of experiments is carried out using Bamboo's simulation mode, enabling us to conduct experiments at a large scale compared to PlanetLab. In the simulation mode, nodes are instantiated based on the pairwise latencies from the `Meridian` dataset, the largest database of pairwise latencies of which we are aware. Figure 8.4 plots the cumulative distribution of median errors (the difference between estimated and actual latencies) and median error ratios (the percentage difference between estimated and actual latencies) achieved by Vivaldi after stabilization for the `Meridian` dataset. In particular, the Figure shows that Vivaldi effectively estimates distances – the system error ratio (the median of the median errors) is just 8.06ms (14.03%). Moreover, 96.80% of nodes have median errors of less than 20ms.

To distribute the burden of bootstrapping peers, nodes join the simulated network at the rate of one node every second until all nodes are present. Nodes join via an already joined peer selected uniformly at random. Simulations were repeated ten times. The results described below reflect cumulative distributions across all iterations.

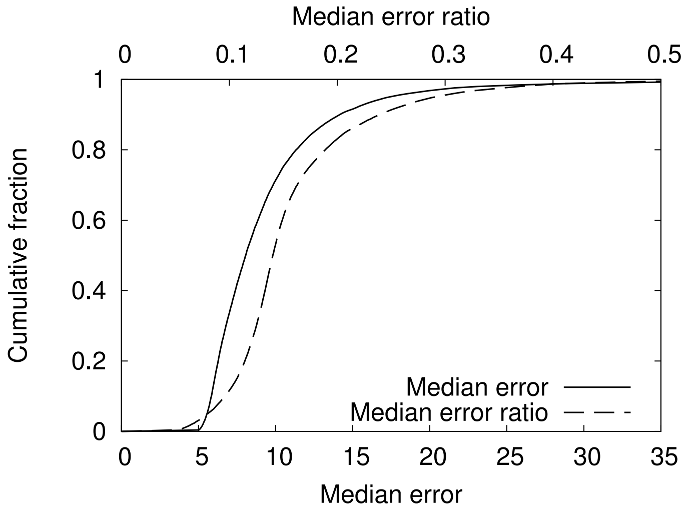


Figure 8.4: The cumulative distribution of median error ratios and median errors (in ms) of the `Meridian` dataset.

Performance Analysis Figure 8.5 shows the CDF of the performance improvements (shown as decreases in RTT) of the Coord and Rand strategies. We make the following observations. First, Coord is more effective than Rand at finding paths with lower RTTs. When one ($k = 1$), three ($k = 3$), or nine ($k = 9$) paths are considered, Coord consistently outperforms Rand. For example, when $k = 5$, Coord decreased the RTT of nearly 60% of paths, reducing the RTT by at least 50ms for 23% of paths and by 100ms for 8% of routes. In comparison, Rand improved the RTT of only 27% of paths. Second, we observe that our performance improvements are consistent with the trace-driven analysis presented in Section 8.2. In the steady state, each Contour node caches the coordinates of 180 other Contour nodes (7% of the network). Contour's improvements in RTT approach those in the optimality analysis presented in Figure 8.2. The optimal analysis performs slightly better as it assumes exact latencies between peers.

Figure 8.6 plots the RTTs measured using the direct route (x-axis) against the

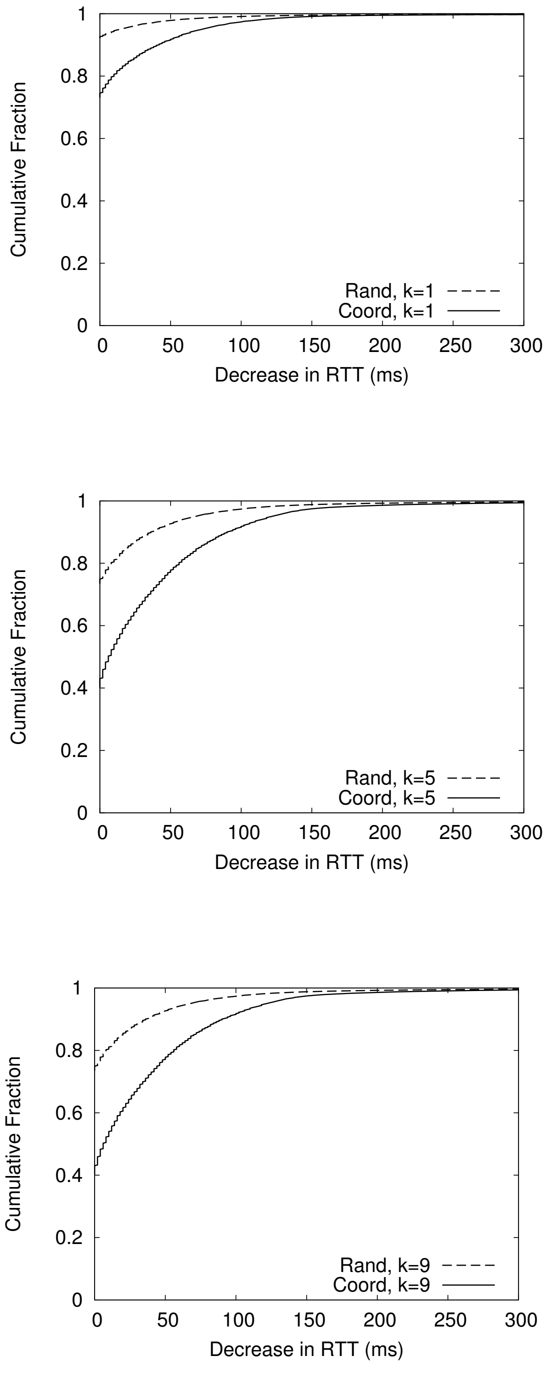


Figure 8.5: Cumulative distribution of the decrease in RTT (in ms) for $k = 1$ *(top)*, $k = 5$ *(middle)*, and $k = 9$ *(bottom)*.

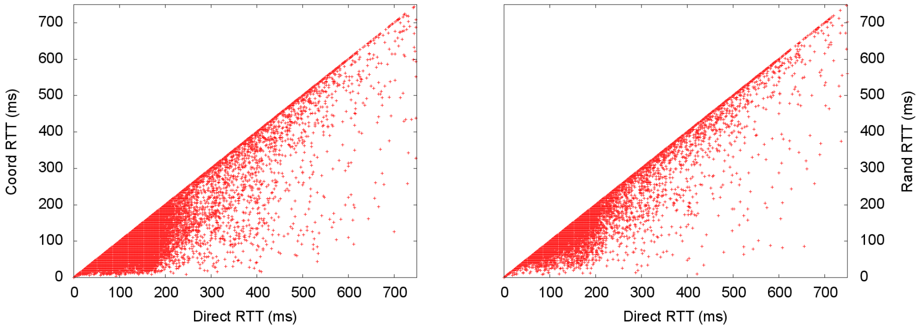
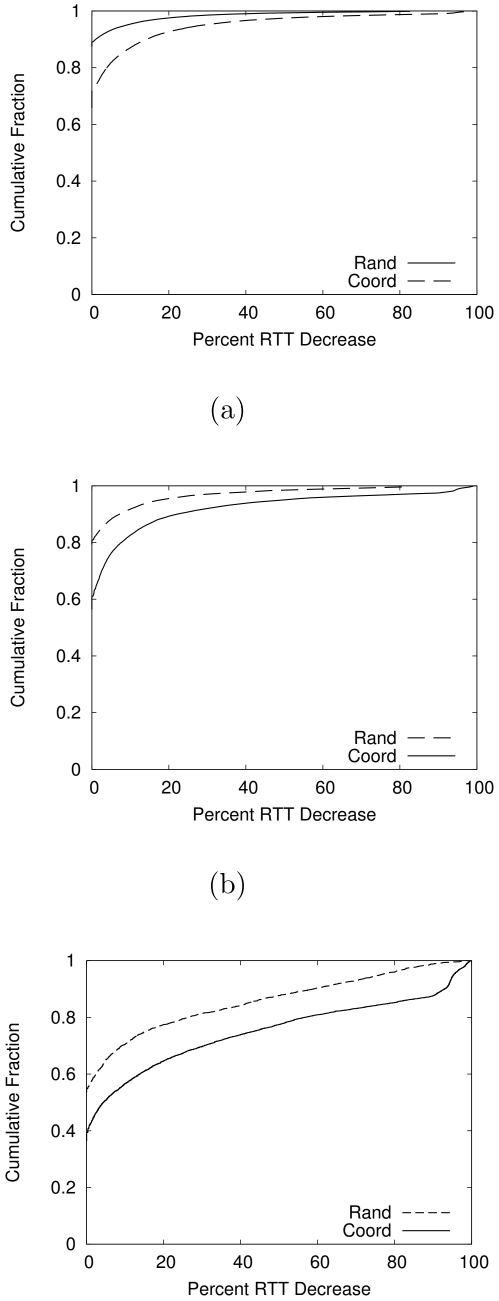


Figure 8.6: The relationship between RTTs measured using the direct path (x-axis) and RTTs measured using Contour (y-axis) for the Coord (left) and Rand (right) strategies, with $k = 5$.

RTTs measured using Coord (left graph) and Rand (right graph) when $k = 5$. Since Contour reverts to direct communication when detour routes do not yield improved performance, no points exist above the identity function. Vertical distance from the identity function denotes the decrease in RTT obtained using Contour. As can be discerned from the bottom right quadrant of the graphs, the Coord strategy is more adept at decreasing RTT for direct routes with otherwise high RTTs. Contour RTT is able to achieve low latency paths ($< 150ms$) even when the corresponding direct RTT path is prohibitively high ($> 450ms$). Our results indicate that Coord not only improves upon existing paths (as reflected in Figure 8.5), but is able to avoid direct paths of high RTTs.

8.4.2 PlanetLab Deployment

To validate our simulation results, we deployed Contour on 163 geographically distributed nodes on the PlanetLab testbed. End-to-end measurements were conducted by the sender and included all processing and network overheads incurred by the relay node. In each experiment, all PlanetLab nodes joined the network within approximately three minutes of the first participant. Our workload consists of establishing a Contour path from a random source to destination every five seconds.



(a)

(b)

(c)

Figure 8.7: (a) Cumulative distribution of percent decreases in RTTs achieved by Contour on PlanetLab for all direct paths, (b) direct paths with RTTs exceeding 150ms, and (c) direct paths with RTTs exceeding 300ms.

Latency As discussed in Section 8.2, the prevalence of TIVs tends to increase with network size, and thus we would expect to see fewer TIVs (and hence worse detour performance) in the significantly smaller PlanetLab network than in the 2500 node Meridian simulation. Nevertheless, the improvements presented below serve to provide a lower-bound on the potential of Contour at Internet-scale.

Figure 8.7a shows the CDF of performance improvements (in terms of percentage decrease in RTT) as a results of Contour detour paths. For both Coord and Rand, RTT measurements are conducted using the best of five detour paths ($k = 5$). In Figures 8.7b and 8.7c, we additionally show a similar CDF of performance improvements when limited to direct paths with RTT of at least $150ms$ and $300ms$ respectively.

We make the following observations. First, Coord performs better than Rand as expected for finding shortest latency paths. 28% of all paths are improved using Coord, compared with 11% when Rand is used. Second, we observe that Contour is highly effective at improving the performance of routes with high RTTs, where 19% (45%) of paths with RTTs of at least $150ms$ ($300ms$) are improved. Coord also outperforms Rand and improves e2e RTT for 39% of the paths that would otherwise have RTTs exceeding $150ms$ and 61% of those that have RTTs beyond 300ms. 25% of the paths with RTTs exceeding $300ms$ experience a 43% reduction in RTT using Coord.

8.5 Summary

This chapter introduces Contour, a fully distributed and scalable detour routing system that utilizes distributed directory services and lightweight coordinate systems to intelligently select detour paths.

Our trace-driven simulation results demonstrate that Contour is highly effective at producing paths that improve upon performance at low overheads. For instance, Contour is able to find paths that decrease the RTT of 20% of the paths by at least 40%. Even on a limited deployment of 163 nodes on PlanetLab testbed, Contour improves the performance of 61% of paths that would otherwise exceed 300ms.

Chapter 9

Conclusion

This dissertation presents an architecture for scalable, decentralized, and tunable anonymity services for the Internet. Our techniques utilize virtualized coordinate systems in which network distances are embedded in n-dimensional Euclidean space. By aggregating the distances between overlay nodes' coordinates, our path selection strategies accurately estimate the e2e performance characteristics of potential paths, empowering applications to select high performance anonymous routes.

Our results indicate that the proposed relay selection strategies are highly effective at producing anonymous paths with low latency, jitter, loss, AS traversals, and high bandwidth. In particular, our Weighted algorithm reduced e2e path latencies by more than half, decreased jitter by nearly 40%, and reduced the number of AS traversals by 16% when compared to selecting relays uniformly at random.

In addition to achieving better performance, we demonstrate that our *link-based* relay selection algorithm provides better anonymity than more traditional *node-based* techniques. For example, when an adversary controls the top 10% of relays as ranked by their available bandwidth, she is able to compromise 55% of anonymous paths on the Tor network when the default node-based Tor relay selection algorithm is used. In contrast, when the attacker conducts the analogous attack against link-based selection (here, controlling the endpoints of the top 10% of links as ranked by *link* bandwidths), the adversary can compromise less than 3% of anonymous routes.

To demonstrate the practicality of coordinate-based anonymous routing, we propose the *Application-Aware Anonymity* (A^3) framework. Unlike most existing anonymity systems, A^3 is fully decentralized, requiring no fixed infrastructure, specialized trusted nodes, public key infrastructure, or *a priori* shared secrets. Our evaluation of A^3 on the PlanetLab testbed reveals that the system is highly scalable, and is estimated to require a modest 9.5KBps of bandwidth per peer in a one million node deployment. Additionally, our PlanetLab results mirror simulation studies, demonstrating that our link-based selection strategy reduces the e2e roundtrip time (RTT) of anonymous paths by 57%.

We further show that coordinate-based routing techniques are appropriate for locating high performance *non-anonymous* paths in overlay networks. To that end, we introduced *Contour*, a detour routing system that leverages coordinate embedding systems to locate overlay paths with e2e performance that exceeds that of direct IP communication. Experiments conducted on a deployed Contour network on PlanetLab show that Contour decreases the RTT of 20% of the paths by at least 40% as compared to direct connections.

The coordinate-based routing techniques used by A^3 and Contour depend on the accuracy of their underlying coordinate systems. Previous work has shown that such systems are vulnerable to insider manipulation [45]. To protect the accuracy of the coordinate system (and consequently, the performance of Contour and A^3), we present *Veracity*, a vote-based protocol that requires that advertised coordinates be verified by an independent set of peers before being used. Veracity is fully decentralized, requiring no fixed infrastructure, making it ideal for anonymity services such as A^3. Our results show that Veracity can effectively mitigate attacks when up to 30% of the network is controlled by malicious nodes.

A^3, Contour, and Veracity demonstrate the feasibility, effectiveness, and security of coordinate-based routing. Coupled together, A^3 and Veracity enable high performance and secure routing on overlay networks, enabling the anonymization of network-intensive services previously deemed too restrictive for anonymity services.

Bibliography

[1] Advanced Network Technology Center, University of Oregon. Route Views. `http://www.routeviews.org/`.

[2] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *3rd ACM Conference on Internet Measurement (IMC)*, 2003.

[3] Madhukar Anand, Eric Cronin, Micah Sherr, Matt Blaze, and Sampath Kannan. Security Protocols with Isotropic Channels. Technical Report TR-CIS-06-18, University of Pennsylvania, Department of Computer and Information Science, November 2006. Available at `http://www.cis.upenn.edu/~msherr/papers/isotropism-tr-cis-06-18.pdf`.

[4] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. The Case for Resilient Overlay Networks. In *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, page 152, 2001.

[5] Marc Sanchez Artigas, Pedro Garcia Lopez, and Antonio F. Gomez Skarmeta. A novel methodology for constructing secure multipath overlays. *IEEE Internet Computing*, 9(6):50–57, 2005.

[6] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking Up Data in P2P Systems. *Communications of the ACM, Vol. 46, No. 2*, February 2003.

[7] The Bamboo Distributed Hash Table. `http://bamboo-dht.org/`.

[8] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, pages 11–20, 2007.

[9] Oliver Berthold, Hannes Federrath, and Marit Köhntopp. Project "Anonymity and Unobservability in the Internet". In *CFP '00: Proceedings of the Tenth Conference on Computers, Freedom and Privacy*, pages 57–65, 2000.

[10] Nikita Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays.* PhD thesis, University of California, Berkeley, 2005.

[11] Nikita Borisov. Computational Puzzles as Sybil Defenses. In *IEEE International Conference on Peer-to-Peer Computing*, pages 171–176, 2006.

[12] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *OSDI*, 2002.

[13] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[14] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[15] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[16] M. Costa, M. Castro, R. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *International Conference on Distributed Computing Systems*, 2004.

[17] Eric Cronin, Micah Sherr, and Matt Blaze. Listen Too Closely and You May Be Confused. In *Thirteenth International Workshop on Security Protocols (SPW)*, 2005.

[18] Eric Cronin, Micah Sherr, and Matt Blaze. On the Reliability of Current Generation Network Eavesdropping Tools. In *Second Annual IFIP WG 11.9 International Conference on Digital Forensics*, January 2006.

[19] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. *SIGCOMM Comput. Commun. Rev.*, 34(4):15–26, 2004.

[20] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-Area Cooperative Storage with CFS. In *SOSP*, 2001.

[21] Frank Dabek, Jinyang Li, Emil Sit, Frans Kaashoek, Robert Morris, and Chuck Blake. Designing a DHT for Low Latency and High Throughput. In *NSDI*, 2004.

[22] George Danezis, Chris Lesniewski-Laas, M. Frans Kaashoek, and Ross Anderson. Sybil-Resistant DHT Routing. *Computer Security – ESORICS 2005*, 2005.

[23] Debian. OpenSSL – Predictable Random Number Generator. Debian Security Advisory DSA-1571-1, Debian, March 2008.

[24] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, Internet Engineering Task Force, January 1999.

[25] Jochen Dinger and Hannes Hartenstein. Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration. In *International Conference on Availability, Reliability and Security*, pages 756–763, 2006.

[26] Roger Dingledine. Personal communication, March 2009.

[27] Roger Dingledine and Nick Mathewson. Tor Path Specification, January 2008. `http://www.torproject.org/svn/trunk/doc/spec/path-spec.txt`.

[28] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proc. of the 13th Usenix Security Symposium*, pages 303–320, 2004.

[29] Roger Dingledine and Steven J. Murdoch. Performance Improvements on Tor or, Why Tor is Slow and What Were Going to Do About It, March 2009. Published as Tor Roadmap. Available at `https://svn.torproject.org/svn/tor/trunk/doc/roadmaps/2009-03-11-performance.pdf`.

[30] Dave Dittrich. tcpdstat. `http://staff.washington.edu/dittrich/talks/core02/tools/tools.html`.

[31] John R. Douceur. The Sybil Attack. In *First International Workshop on Peer-to-Peer Systems*, March 2002.

[32] William Enck, Patrick Traynor, Patrick McDaniel, and Thomas La Porta. Exploiting Open Functionality in SMS-Capable Cellular Networks. In *CCS '05: Proceedings of the 12th ACM Conference on Computer and Communications Security*, pages 393–404, New York, NY, USA, 2005. ACM Press.

[33] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *18th USENIX Security Symposium*, August 2009.

[34] Nick Feamster and Roger Dingledine. Location Diversity in Anonymity Networks. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 66–76, 2004.

[35] Hannes Federrath. JAP: Anonymity & Privacy. `http://anon.inf.tu-dresden.de/`.

[36] Amos Fiat, Jared Saia, and Maxwell Young. Making Chord Robust to Byzantine Attacks. In *In Proc. of the European Symposium on Algorithms*, 2005.

[37] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *CCS*, Washington, D.C., November 2002.

[38] M.J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *Networked Systems Design and Implementation (NSDI)*, May 2006.

[39] Thomer M. Gil, Frans Kaashoek, Jinyang Li, Robert Morris, and Jeremy Stribling. p2psim: A Simulator for Peer-to-Peer Protocols. `http://pdos.csail.mit.edu/p2psim/`.

[40] Corrado Gini. Measurement of Inequality of Incomes. *The Economic Journal*, 31(121):124–126, March 1921.

[41] Krishna P. Gummadi, Harsha Madhyastha, Steven D. Gribble, Henry M. Levy, and David J. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *OSDI*, 2004.

[42] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *ACM SIGCOMM Workshop on Internet Measurment (IMW)*, 2002.

[43] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from Anonymity. In *IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 239–248, 2006.

[44] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kave Salamatian, Thierry Turletti, and Walid Dabbous. Securing Internet Coordinate Embedding Systems. In *ACM SIGCOMM*, August 2007.

[45] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Real Attacks on Virtual Networks: Vivaldi Out of Tune. In *SIGCOMM Workshop on Large-Scale Attack Defense (LSAD)*, pages 139–146, 2006.

[46] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet.* Scribner, 1996.

[47] "King" Data Set. Available at `http://pdos.csail.mit.edu/p2psim/kingdata/`.

[48] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. *SIGCOMM Comput. Commun. Rev.*, 30(4):175–187, 2000.

[49] K. Lakshminarayanan, I. Stoica, and S. Shenker. Routing as a Service. Technical Report UCB-CS-04-1327, UC Berkeley, 2004.

[50] Karthik Lakshminarayanan and Venkata N. Padmanabhan. Some Findings on the Network Performance of Broadband Hosts. In *3rd ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 45–50, 2003.

[51] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[52] Jonathan Tormod Ledlie. *A Locality-Aware Approach to Distributed Systems.* PhD thesis, Harvard University, September 2007.

[53] Sanghwan Lee, Zhi-Li Zhang, Sambit Sahu, and Debanjan Saha. On Suitability of Euclidean Embedding of Internet Hosts. *SIGMETRICS Perform. Eval. Rev.*, 34(1):157–168, 2006.

[54] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi. Constructing Internet Coordinate System Based on Delay Measurement. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 129–142, New York, NY, USA, 2003. ACM.

[55] Eng Keong Lua, Timothy G. Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft. On the Accuracy of Embeddings for Internet Coordinate Systems. In *Internet Measurment Conference*, 2005.

[56] C. Lumezanu, D. Levin, and N. Spring. PeerWise Discovery and Negotiation of Shorter Paths. In *Workshop on Hot Topics in Networks (HotNets)*, 2007.

[57] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. IPlane: An Information Plane for Distributed Services. In *7th Symposium on Operating Systems Design and Implementation (OSDI '06)*, 2006.

[58] MaxMind GeoIP City Database. `http://www.maxmind.com/app/city`.

[59] Petar Maymounkov and David Mazires. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[60] D. McCoy, K. Bauer, D. Grunwald, P. Tabriz, and D. Sicker. Shining Light in Dark Places: A Study of Anonymous Network Usage. In *8th Privacy Enhancing Technologies Symposium (PETS 2008)*, July 2008.

[61] Mixmaster. `http://mixmaster.sourceforge.net/`.

[62] Mervin E. Muller. A Note on a Method for Generating Points Uniformly on n-dimensional Spheres. *Communications of the ACM*, 2(4):19–20, 1959.

[63] Steven J. Murdoch. Hot or Not: Revealing Hidden Services by Their Clock Skew. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 27–36, 2006.

[64] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy*, pages 183–195, 2005.

[65] Steven J. Murdoch and Robert N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *8th Privacy Enhancing Technologies Symposium (PETS 2008)*, July 2008.

[66] T. S. Eugene Ng and Hui Zhang. A Network Positioning System for the Internet. In *Proceedings of the 2004 USENIX Annual Technical Conference*, June 2004.

[67] CW O'Donnell and V. Vaikuntanathan. Information Leak in the Chord Lookup Protocol. In *Fourth International Conference on Peer-to-Peer Computing*, pages 28–35, August 2004.

[68] Lasse Øverlier and Paul Syverson. Locating Hidden Servers. In *IEEE Symposium on Security and Privacy*, 2006.

[69] Vern Paxson. End-to-End Routing Behavior in the Internet. *SIGCOMM Comput. Commun. Rev.*, 36(5):41–56, 2006.

[70] Fabien A.P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Information Hiding – A Survey. *Proceeding of the IEEE, special issue on protection of multimedia context*, 1999.

[71] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 1–9, 2000.

[72] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-MIXes: Untraceable Communication with Small Bandwidth Overhead. In *Proceedings of Kommunikation in Verteilten Systemen, Grundlagen, Anwendungen, Betrieb, GI/ITG-Fachtagung, Mannheim, 20.-22.*, volume 267, pages 451–463, February 1991.

[73] Peter Pietzuch, Jonathan Ledlie, Michael Mitzenmacher, and Margo Seltzer. Network-Aware Overlays with Network Coordinates. In *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*, page 12, Washington, DC, USA, 2006. IEEE Computer Society.

[74] PlanetLab. `http://www.planet-lab.org`.

[75] J. B. Postel. Internet Control Message Protocol. RFC 792, Internet Engineering Task Force, 1981.

[76] J. B. Postel. Transmission Control Protocol. RFC 793, Internet Engineering Task Force, 1981.

[77] Kevin Poulsen. New caller i.d. spoofing site opens, October 2004. `http://www.securityfocus.com/news/9822`.

[78] Thierry Rakotoarivelo, Patrick Senac, Aruna Seneviratne, and Michel Diaz. A Structured Peer-to-Peer Method to Discover QoS Enhanced Alternate Paths. In *Third International Conference on Information Technology and Applications*, 2005.

[79] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Ittai Abraham, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. A unified network coordinate system for bandwidth and latency. Technical Report MSR-TR-2008-124, Microsoft Research, September 2008.

[80] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. On the Treeness of Internet Latency and Bandwidth. In *SIGMETRICS/Performance*, June 2009.

[81] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM 2001*, August 2001.

[82] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4), May 1998.

[83] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. In *ACM Transactions on Information and System Security*, 1998.

[84] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection. In *WPES '02: Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society*, pages 91–102, New York, NY, USA, 2002. ACM Press.

[85] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. In *USENIX Technical Conference*, June 2004.

[86] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Passive and Active Measurement Workshop*, 2003.

[87] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware*, pages 329–350, 2001.

[88] Damien Saucez. Securing Network Coordinate Systems. Master's thesis, Universit catholique de Louvain, 2007.

[89] Damien Saucez, Benoit Donnet, and Olivier Bonaventure. A Reputation-Based Approach for Securing Vivaldi Embedding System. In *Dependable and Adaptable Networks and Services*, 2007.

[90] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: a Case for Informed Internet Routing and Transport. *IEEE Micro*, 19(1):50–59, 1999.

[91] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The End-to-end Effects of Internet Path Selection. *SIGCOMM Comput. Commun. Rev.*, 29(4):289–299, 1999.

[92] A. Serjantov and P. Sewell. Passive-Attack Analysis for Connection-Based Anonymity Systems. *International Journal of Information Security*, 4(3):172–180, 2005.

[93] Y. Shavitt and T. Tankel. Big-bang Simulation for Embedding Network Distances in Euclidean Space. In *IEEE Infocom*, April 2003.

[94] Micah Sherr, Matt Blaze, and Boon Thau Loo. Contour: Coordinate-Based Detour Routing, 2009. *In preparation.*

[95] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable Link-Based Relay Selection for Anonymous Routing. In *9th Privacy Enhancing Technologies Symposium (PETS 2009)*, August 2009.

[96] Micah Sherr, Matt Blaze, and Boon Thau Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *USENIX Annual Technical Conference (USENIX '09)*, June 2009.

[97] Micah Sherr, Eric Cronin, and Matt Blaze. Measurable Security Through Isotropic Channels. In *Fifteenth International Workshop on Security Protocols*, April 2007.

[98] Micah Sherr, Eric Cronin, Sandy Clark, and Matt Blaze. Signaling Vulnerabilities in Wiretapping Systems. *IEEE Security & Privacy*, 3(6):13–25, November 2005.

[99] Micah Sherr, Boon Thau Loo, and Matt Blaze. Towards Application-Aware Anonymous Routing. In *Second USENIX Workshop on Hot Topics in Security (HotSec)*, August 2007.

[100] Micah Sherr, Boon Thau Loo, and Matt Blaze. Veracity: A Fully Decentralized Service for Securing Network Coordinate Systems. In *IPTPS*, February 2008.

[101] Clay Shields and Brian Neil Levine. A Protocol for Anonymous Communication over the Internet. In *CCS '00: Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 33–42, New York, NY, USA, 2000. ACM Press.

[102] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[103] Robin Snader and Nikita Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *15th Annual Network and Distributed System Security Symposium (NDSS)*, February 2008.

[104] Speakeasy Speed Test. `http://www.speakeasy.net/speedtest/`.

[105] Frank Stajano and Ross J. Anderson. The Cocaine Auction Protocol: On the Power of Anonymous Broadcast. In *Information Hiding*, pages 434–447, 1999.

[106] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, 2001.

[107] Tor Project. Tor: Anonymity Online. `http://www.torproject.org/`.

[108] Vitelity communications, inc. `http://vitelity.com`.

[109] Vuze Bittorrent Client. `http://azureus.sourceforge.net/`.

[110] D.S. Wallach. A Survey of Peer-to-Peer Security Issues. *Lecture Notes in Computer Science*, pages 42–57, 2003.

[111] Guohui Wang, Bo Zhang, and T. S. Eugene Ng. Towards Network Triangle Inequality Violation Aware Distributed Systems. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 175–188, New York, NY, USA, 2007. ACM.

[112] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirecion on CDN Robustness. In *OSDI*, 2002.

[113] Matt Welsh, David Culler, and Eric Brewer. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. *SIGOPS Oper. Syst. Rev.*, 35(5):230–243, 2001.

[114] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a Lightweight Network Location Service without Virtual Coordinates. In *SIGCOMM*, 2005.

[115] Bernard Wong, Alksandrs Slivkins, and Emin Gun Sirer. ClosestNode.com. `http://www.closestnode.com/`.

[116] Matthew Wright, Micah Adler, Brian Levine, and Clay Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004.

[117] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.J. Lee. S^3: a Scalable Sensing Service for Monitoring Large Networked Systems. In *SIGCOMM Internet Network Management Workshop*, 2006.

[118] D. J. Zage and C. Nita-Rotaru. On the Accuracy of Decentralized Virtual Coordinate Systems in Adversarial Networks. In *CCS*, 2007.

[119] Sebastian Zander and Steven J. Murdoch. An improved clock-skew measurement technique for revealing hidden services. In *17th USENIX Security Symposium*, pages 211–225, July 2008.

[120] Bo Zhang, T. S. Eugene Ng, Animesh Nandi, Rudolf Riedi, Peter Druschel, and Guohui Wang. Measurement Based Analysis, Modeling, and Synthesis of the Internet Delay Space. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 85–98, 2006.

[121] H. Zheng, E.K. Lua, M. Pias, and T. Griffin. Internet Routing Policies and Round-trip-times. In *Proceedings of the Passive Active Measurement.* Springer, 2005.