An Information Plane for Internet Applications

Harsha V. Madhyastha

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2008

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Harsha V. Madhyastha

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

Thomas E. Anderson

Arvind Krishnamurthy

Reading Committee:

Thomas E. Anderson

Arvind Krishnamurthy

Arun Venkataramani

Date:

University of Washington

Abstract

An Information Plane for Internet Applications

Harsha V. Madhyastha

Co-Chairs of the Supervisory Committee:
Professor Thomas E. Anderson
Department of Computer Science and Engineering

Research Assistant Professor Arvind Krishnamurthy
Department of Computer Science and Engineering

Over the last few years, several new applications have emerged on the Internet that are distributed at a scale previously unseen. Examples include peer-to-peer filesharing, content distribution networks, and voice-over-IP. This new class of distributed applications can make intelligent choices among the several paths available to them to optimize their performance. However, as a result of the best-effort packet forwarding interface exported by the Internet, applications need to explicitly measure the network to discover information about any path. Not only does measurement at the time of communication impose a significant overhead on most applications, but it is also redundant to have every application reimplement an Internet measurement component.

In this dissertation, I design, build, and evaluate an information plane for the Internet, called iPlane, that enables distributed applications to discover information about Internet paths without explicit measurement. iPlane efficiently performs measurements from end-hosts under its control to predict path properties on the Internet between arbitrary end-hosts. I pursue a structural approach in issuing and synthesizing measurements—instead of using only end-to-end measurements and thus treating the Internet as a blackbox, I discover the Internet's routing topology and then compose measurements of links and path segments. This structural approach enables iPlane to predict multiple path properties, such as latency

and loss rate.

My evaluation of iPlane shows that iPlane's predictions of paths and path properties are accurate, significantly better than previous approaches for some of the sub-problems that iPlane tackles. Also, I used information from iPlane to drive path selection in three representative distributed applications—content distribution, peer-to-peer filesharing, and voice-over-IP. In each case, the use of iPlane helped significantly improve application performance.

TABLE OF CONTENTS

Page

List of Figures . . . v

List of Tables . . . vii

Chapter 1: Introduction . . . 1
- 1.1 Reasons to Build an Information Plane . . . 3
  - 1.1.1 An Information Plane is Better than a Library . . . 3
  - 1.1.2 Make Information Available Now, not Later . . . 4
  - 1.1.3 Can Leverage Work on Measuring the Internet . . . 5
- 1.2 Challenges and Goals . . . 5
  - 1.2.1 Predictions between Arbitrary End-Hosts: Internet-wide coverage . . . 6
  - 1.2.2 Accurate estimation of multiple path properties: Topology-aware and Accurate . . . 6
  - 1.2.3 Minimizing measurement traffic: Efficient and Unobtrusive . . . 8
  - 1.2.4 Compatible with the growth of measurement vantage points: Scalable 8
  - 1.2.5 Summary . . . 9
- 1.3 Thesis and Contributions . . . 9
- 1.4 Organization . . . 10

Chapter 2: Background and Motivation . . . 12
- 2.1 Internet Topology . . . 12
- 2.2 Routing in the Internet . . . 14
- 2.3 Path Properties . . . 17
- 2.4 Need for an Information Plane . . . 19

Chapter 3: Related Work . . . 20
- 3.1 Internet Measurement . . . 20
  - 3.1.1 Measurement tools and techniques . . . 20
  - 3.1.2 Measurement studies . . . 24
  - 3.1.3 Measurement systems . . . 25

3.2 Prior Work towards an Information Plane . . . 27
3.3 Overlay Systems . . . 29

Chapter 4: Path Prediction by Composing Paths . . . 31
4.1 Overview . . . 31
4.2 Goals . . . 34
4.3 Building an Atlas . . . 34
4.4 Path Composition . . . 36
4.5 Clustering Routers . . . 38
4.5.1 Alias resolution . . . 39
4.5.2 Intra-AS clustering . . . 40
4.6 Path Selection . . . 45
4.6.1 Modeling Default Routing Policy . . . 45
4.6.2 Export Policies . . . 46
4.7 Optimization of Atlas Size . . . 47
4.8 Evaluation . . . 47
4.8.1 Measurements . . . 48
4.8.2 Path Composition: Feasibility Analysis . . . 48
4.8.3 Path Prediction . . . 51
4.9 Summary . . . 52

Chapter 5: Graph-based Path Prediction . . . 54
5.1 The Problem: Modeling Internet Routing . . . 55
5.2 Graph: A first cut . . . 56
5.3 Sources of Prediction Error . . . 61
5.4 Addressing asymmetry . . . 62
5.5 Export Policies . . . 63
5.6 AS preferences . . . 64
5.7 Incorporating traffic engineering . . . 65
5.8 Evaluation . . . 66
5.8.1 Measurement data . . . 66
5.8.2 Can the algorithm predict AS paths accurately? . . . 68
5.8.3 Does the atlas scale with end-host measurements? . . . 69
5.8.4 How stationary is the atlas over time? . . . 70
5.8.5 Improved Path Composition Technique . . . 71
5.9 Summary . . . 72

Chapter 6: Measuring Link Metrics . . . 74
6.1 Overview . . . 74
6.2 Measuring path segment latency . . . 76
6.3 Measuring link latency . . . 77
6.3.1 Identifying symmetric links in routes . . . 78
6.3.2 Measuring link latencies using spoofed probes . . . 80
6.3.3 Putting latency samples together . . . 82
6.4 Measurement of other link attributes . . . 83
6.4.1 Measuring Loss Rate . . . 84
6.4.2 Orchestrating the Measurement Tasks . . . 85
6.5 Evaluation . . . 86
6.5.1 Accuracy of link latency estimates . . . 87
6.5.2 Accuracy of latency estimates for known paths . . . 88
6.5.3 Accuracy of latency estimates for predicted paths . . . 89
6.5.4 Accuracy of path loss rate estimates . . . 90
6.5.5 Accuracy of application-level metrics . . . 91
6.6 Summary . . . 94

Chapter 7: Implementation and Evaluation of iPlane . . . 97
7.1 iPlane: An Information Plane . . . 97
7.2 Evaluation . . . 98
7.2.1 Stationarity of Measurements . . . 98
7.2.2 Scalability . . . 100
7.3 Query Interface . . . 102
7.4 Application Case Studies . . . 104
7.4.1 BitTorrent . . . 104
7.4.2 Voice-over-IP . . . 106
7.4.3 Content Distribution Network . . . 108
7.5 Summary . . . 108

Chapter 8: Conclusions and Future Work . . . 110
8.1 Thesis and Contributions . . . 110
8.2 Key Ideas . . . 111
8.3 Future Work . . . 112
8.3.1 Improving iPlane's Accuracy . . . 113
8.3.2 Building Internet-scale Distributed Applications . . . 113

8.3.3 Scaling iPlane 114
8.3.4 Longitudinal Analysis of the Internet 116
8.4 Summary 116

Bibliography 117

LIST OF FIGURES

Figure Number Page

2.1 Internet's structure comprising end-hosts and autonomous systems. 13
2.2 Structure of an autonomous system. 14
2.3 Illustrative example demonstrating early-exit routing. 16

3.1 Sample output of traceroute. 22

4.1 Illustration of path prediction by composing path segments. 37
4.2 Comparison of reverse path lengths of interfaces determined to be aliases. . . 42
4.3 Comparison of reverse path lengths of interfaces within an AS. 44
4.4 Feasibility of AS path prediction with path composition. 50
4.5 Comparison of different path prediction policies to optimal. 51

5.1 The Graph algorithm to predict a route from s to d in a graph G. 58
5.2 Enforcement of customer>peer>provider in the Graph algorithm. 60
5.3 Division of atlas into two planes to account for routing asymmetry. 63
5.4 Example illustrating components of graph-based path prediction. 65
5.5 Evaluation of graph-based path prediction. 68
5.6 Increase in size of graph-based atlas with addition of end-host measurements. 69
5.7 Change in the set of links in graph-based atlas over time. 71
5.8 Path prediction with path composition augmented with routing policy. 72

6.1 Procedure to identify symmetry using traceroute and IP record-route. 78
6.2 Procedure to identify symmetry using TTL-limited UDP probes. 78
6.3 Procedure to identify symmetry using the IP timestamp option. 79
6.4 Procedure to measure clock difference between vantage points. 80
6.5 Computing the clock difference between nodes. 81
6.6 Procedure to measure one-way latency. 83
6.7 Spoofing technique to measure one-way latency. 84
6.8 Comparison of spread in link latency samples. 87
6.9 Accuracy of path latency estimates on known paths with respect to coverage. 88
6.10 Accuracy of end-to-end latency estimates for predicted paths. 89

6.11 Accuracy of end-to-end loss rate estimates for predicted paths. 91
6.12 Accuracy of identifying closest 10 destinations using predicted latency. 92
6.13 Evaluation of detours chosen using predicted latencies. 93
6.14 Rank correlation coefficient between measured and predicted TCP throughput. 95

7.1 Stationarity of measurements over different intervals over the course of a day. 99
7.2 Evaluation of the use of iPlane to perform informed peer selection in BitTorrent. 105
7.3 Evaluation of the use of iPlane to choose relay nodes in Voice-over-IP. 107
7.4 Evaluation of the use of iPlane to choose replicas in a CDN. 109

LIST OF TABLES

Table Number Page

4.1 Summary of techniques used to predict paths by composing path segments. . 33

5.1 Summary of techniques used to predict paths from an atlas of links. 57

5.2 Breakdown of sizes of different components of graph-based atlas. 67

6.1 Summary of techniques used to measure link metrics. 75

7.1 Complexity of measurement techniques used in iPlane. 101

ACKNOWLEDGMENTS

I am extremely fortunate to have had the opportunity to work with such a fantastic team of advisors. Tom's vision for the big picture combined with his keen eye for the minutest detail has been the basis for much of the work in this dissertation. Arvind's never-ending stream of ideas and insatiable enthusiasm have been my primary source of motivation. Arun has been a sounding board that I could always rely upon to judge my ideas and his help in choosing and prioritizing among the several options that one invariably has in research has been invaluable. I can only dream of aspiring to combine their enthusiasm, judgment, and intellect in my career.

Many portions of this dissertation are the result of work I have done with other members of the networks group. Colin Dixon, Tomas Isdal, Ethan Katz-Bassett, and Michael Piatek have all contributed in my work towards building and evaluating iPlane. The four of them, along with Ivan Beschastnikh, Daniel Halperin, and John P. John, have also consistently provided me with great feedback on all the talks I have given on this work.

Shannon Gilmore, Melody Kadenko, and Julie Svendsen ensured that I never had to worry about administrative details. Lindsay Michimoto always had a ready answer to any question I threw at her all through graduate school. I am immensely thankful to all of them for their help.

A number of students at UW ensured that I had a life outside of work but none more so than Nilesh Dalvi and Sumit Sanghai. Nilesh showed me that it is possible to consistently do great research even while having fun. I doubt I will ever meet anyone else in my life with whom I would be able to share my passion for sports as I did with Sumit. I will always fondly look back on the times I spent with them.

Most importantly, my parents and my brother have been the pillars of my life. They have been there for me whenever I needed emotional support. They have always trusted my judgment and yet have been a source of guidance I could always fall back on. I have no hope of paying them back for their thankless love and support.

Lastly, I thank the Almighty for making me physically and mentally capable of gaining admission to a great institution like the University of Washington and to produce this work here. I hope He will provide me with the wisdom to appreciate the fortune I have had compared to a large fraction of the world's population who are less fortunate.

Chapter 1
INTRODUCTION

Over the last few years, several new applications have emerged on the Internet that are distributed at a scale previously unseen. Peer-to-peer filesharing systems, such as BitTorrent [16], eMule [23], and Kazaa [42], have enabled millions of users to share files with each other. Content distribution networks like Akamai [1] replicate content across thousands of servers distributed worldwide. Voice-over-IP applications like Skype [77] enable any arbitrary pair of users to talk to each other over the Internet. All of these applications are in stark contrast to the typical client-server architecture where a client communicates with a single server or with one among a cluster of servers at a given location.

There are a common set of features that characterize this new class of large scale distributed applications.

- These applications typically have one-to-many or many-to-many communication patterns. So, there exist multiple paths over which any communication can be performed. For example, multiple peers can have identical pieces of a file in BitTorrent and multiple Akamai servers might be hosting the same content.

- The option of multiple communication paths permits the choice of a good path based on prevalent network conditions. However, since the Internet exports only a best-effort packet forwarding interface, an application needs to explicitly measure the network to discover the properties of a path. The typical scale of distributed applications implies that the number of candidate paths can be large and so, measuring all of them would be infeasible.

- An application's metric for determining a good path for communication is typically a combination of several network-level metrics. For example, in a content distribution network, the replica that is best for a client is determined by the TCP throughput between the client and the replica, and TCP throughput depends on latency, loss rate, and available bandwidth. While latency could possibly be measured when the communication is initiated, measuring other properties such as loss rate and bandwidth requires several packets to be sent along a path. The measurement overhead can nullify the benefits of choosing a good replica.

- Further, the communication path between a pair of end-hosts may traverse other end-hosts over which they have no control. For example, Skype enables calls between a pair of end-hosts, both behind NATs, by routing the calls through a third end-host that serves as a relay node. The choice of relay node can largely determine end-to-end performance [68]. Thus, the two ends of the call need to be able to estimate properties of the path through each candidate relay node, even without being able to issue measurements from the relay nodes.

All of these application characteristics imply that the new generation of distributed applications can benefit from having *a priori* access to information about the properties of Internet paths—information that the Internet currently does not provide.

In this dissertation, my goal is to bridge the information gap between what the Internet provides and what large scale distributed applications need. I present work that I have done to develop, build, and evaluate an information plane for the Internet, called iPlane. iPlane can provide information about the properties of the path through the Internet between any two arbitrary end-hosts. Applications can query iPlane for this information, and use it to optimize their performance. Conventional client-server applications too can benefit from information about the properties of the path between the client and server, e.g., the bitrate of a video stream can be reduced if the available bandwidth is low. However, I restrict my focus in this dissertation to large-scale distributed applications.

The problem of providing information about path properties to applications without explicit measurement has been studied previously. There has been work in this direction for providing estimates of both latency [24, 56, 18] and loss rate [13, 107]. The techniques for estimating latency typically try to model the Internet as a coordinate space. Every end-host is assigned a coordinate in an Euclidean metric space and the latency between a pair of end-hosts is computed as the distance between their coordinates. While this abstraction is simple and elegant, it is fundamentally handicapped in modeling latency detours [73] because triangle inequality is not violated in an Euclidean metric space. This intrinsically limits the accuracy with which latency can be estimated. On the other hand, techniques to infer loss rates are not scalable as they rely on being able to gather measurements from a number of end-hosts with the same order as the total number of end-hosts in the Internet.

In contrast to the above solutions that treat the Internet as a blackbox and rely only on end-to-end measurements, I pursue a structural approach to estimate the properties of unmeasured paths. From several hundred vantage points distributed around the Internet, a number several orders of magnitude smaller than the number of end-hosts, I issue measurements to discover the Internet's routing topology and infer the properties of links and path segments in the measured structure. I synthesize these measurements to build an annotated *atlas* of the Internet. I then compose relevant parts of the atlas and apply inferred routing policy to estimate the route as well as the path properties between any arbitrary pair of end-hosts. Any end-host can further improve the accuracy of estimates for paths from it to others by augmenting the atlas with a small number of local measurements. The system I build, iPlane repeats the above process to keep the information up-to-date and exports an interface by which applications can query for the properties of paths between any arbitrary pair of end-hosts.

1.1 Reasons to Build an Information Plane

Before I present the challenges in building an information plane and the techniques I have developed in doing so, I first answer the following question: *why build an Internet-wide information plane?*

1.1.1 An Information Plane is Better than a Library

An alternative approach to building an information plane would be to put together a library that applications can link. This library would essentially be a measurement toolkit. When applications request for the properties of any particular path, the library would trigger appropriate measurements of the Internet to discover these properties. Though this would solve the need for information about the network by applications, having each application independently discover the network's state would be an inefficient solution.

On the other hand, a single information plane for the Internet can optimize for both measurements and query processing workload across applications. Measurements and processing necessary to determine the properties of any particular (source, destination) path need be performed only once, instead of once for each application. Moreover, an information plane can take advantage of commonalities of workloads across applications. For example, it can selectively refresh its information about paths used by several applications by triggering relevant measurements more frequently. More generally, an information plane can assimilate measurements made on behalf of all of its clients as well as incorporate information reported by clients to develop a more comprehensive model of Internet behavior over time.

1.1.2 Make Information Available Now, not Later

My approach in building an information plane is to orchestrate measurements of the network from a subset of end-hosts, and use these measurements to make predictions of properties on unmeasured paths. I take this approach, as opposed to proposing changes to the Internet's design, because my goals are practical rather than visionary.

Several modifications to the Internet's architecture have been proposed in the last two decades to improve it along several dimensions, such as scalability [99], security [101], inflexibility due to middleboxes [93], and quality-of-service [6, 5]. Almost all of these proposals have not seen the light of day. The reason for failure of most proposals for change is that any path through the Internet typically traverses multiple Internet Service Providers (ISPs), and the benefit of the new architecture can be realized only by cooperation amongst these competing ISPs. A lack of incentive to be the first-mover prevents any ISP from incorporating the new architecture into its network. Hence, a new Internet architecture proposal that enables the network to export information about its state to applications is unlikely to find favor with ISPs.

iPlane, the result of this dissertation, is a real system that applications on the Internet can use right now. This system has been built with neither any modification to the network, nor by having the network provide more information than what it previously did. Updates from the network, e.g., when any significant change occurs, can help iPlane maintain more up-to-date information about the Internet. However, the system I present in this dissertation does not rely on such information.

1.1.3 Can Leverage Work on Measuring the Internet

To discover information about the network's state from end-hosts, I leverage the large body of work done in the area of Internet measurement. Ever since Paxson's seminal work [63] on analyzing Internet routing, researchers have built a slew of tools that help end-hosts discover properties of the network. Protocols used in the Internet include several features to enable debugging and troubleshooting. Most Internet measurement tools exploit these features in novel ways to discover information that these features were not originally intended to reveal, such as bandwidth capacity. However, use of Internet measurement tools at scale requires careful design, else they can appear as attacks [82]. As a consequence, the extent to which developers of distributed applications have tapped work done by the Internet measurement community has been limited.

Building a single information plane for all applications helps harness all work on Internet measurement in a single system. Having each application developer worry about how to measure the network is both redundant as well as inefficient. Instead, an information plane provides applications with a simple interface—they only need to issue queries to discover information. Moreover, the information plane can ensure that the latest measurement tools are used as tools evolve.

1.2 Challenges and Goals

In this section, I lay down the various challenges associated with building an Internet-wide information plane. These challenges arise because of the Internet's current design and implementation, and because of practical considerations in building a real system that can be widely used. Addressing these challenges are implicitly the goals of my work.

1.2.1 Predictions between Arbitrary End-Hosts: Internet-wide coverage

To be able to serve the needs of large scale distributed applications on the Internet, it is essential for the information plane to have information about the path through the Internet between *any* pair of end-hosts. Each application can run on any arbitrary subset of all the end-hosts in the Internet, and realistically, every application developer's goal is to have this subset constantly growing in size. Therefore, if the information plane provides information for only some fixed set of paths, its value to applications is limited. The application developer then has to build into the application the ability to discover information about paths not covered by the information plane. In such a case, the developer might as well discover information about all paths used by the application, rather than make use of the information plane. Thus, the adoption of the information plane can be severely hurt if it does not provide Internet-wide coverage.

The goal of Internet-wide coverage is however made tougher by the practical restriction that the information plane work without having control over every end-host in the Internet. As of today, the primary way to get a piece of software on to almost all end-hosts is to have it become an essential component of all widely-used operating systems or to have it be bundled with a popular software package such as a web browser. However, any piece of code will be incorporated into operating systems or widely used software packages only if that code is deemed vital, e.g., the TCP/IP networking stack. If I were to go down such a route, my goal of building an information plane will likely reach fruition only after several years. For the same reasons outlined above on why I do not propose changes to the Internet's design, I do not take this approach. Therefore, I need the information plane to provide information between any arbitrary pair of end-hosts without possibly having control over either the source or the destination, i.e., be able to make predictions where measurement is infeasible.

1.2.2 Accurate estimation of multiple path properties: Topology-aware and Accurate

My objective in building an information plane is to provide information about the network to applications, so that applications can use this information to choose between paths.

Therefore, information provided by the information plane would not be of much use if it were not accurate. Further, the information plane must be able to provide estimates of multiple path properties along any given path. This requirement is because of the following two reasons.

1. Performance of different applications can be impacted by different path properties, e.g., download times for short Web transfers are driven by latency, whereas download times for bulk updates are driven by bandwidth.

2. Application performance is typically determined by a combination of path properties, e.g., TCP performance [61] depends on both latency and loss rate.

This need for accurate information on multiple path properties is compounded by the previous goal of being able to provide information on paths that cannot be measured. The problem of using measured path properties to estimate properties along unmeasured paths has received much attention. However, each piece of work focuses on a particular path property, e.g., latency [56, 76], or available bandwidth [33], and how any of these can be extended to infer other path properties is not apparent. The lack of extensibility of existing approaches to infer path properties is because they do not model key aspects of Internet routing. For examples, a popular approach to estimate latency is to embed end-hosts in the Internet into a Euclidean coordinate space, an approach that is not extensible to loss rate.

Instead, it is desirable to have a structural model of the Internet that synthesizes performance predictions based on network topology and routing. Such a model has many strengths by virtue of being able to capture and incorporate the effects of the underlying network structure. A structural model could be used to determine the effect of network topology changes, is likely to be more robust to failures, can be used to pinpoint causes of reduced levels of performance, suggest alternate detour routes, and assist in application server placement. It is also amenable to gradual and further refinement: for example, a disagreement between the path latency estimate provided by the model and observed path latency may trigger further measurement to refine the model.

It is however rather challenging to develop a structural model with good predictive abilities. All of the above mentioned strengths stem from having access to details of the internals of the Internet. As it is both infeasible and unscalable to obtain fine-grained data about the entire Internet, one has to operate with incomplete and coarse-grained information without compromising the predictive abilities of the model. I seek to strike the right balance between scalability and accuracy.

1.2.3 Minimizing measurement traffic: Efficient and Unobtrusive

The information plane is unlikely to remain in operation for long if the measurement traffic it sources significantly impacts normal application traffic. If the measurement traffic imparts severe load on the network, this traffic will soon be filtered by network administrators. Therefore, it is critical that the rates of measurement traffic both arising from any end-host that the information plane uses for measurement and targeted at any particular network element be low. This translates into the requirement that the information plane smooth out its measurement traffic over time. However, the goal of providing Internet-wide coverage and the need to constantly refresh information about the network due to the flux in the Internet's characteristics over time make the orchestration and synthesis of Internet-wide measurements non-trivial.

Apart from the need to ensure that the information plane is not obtrusive, minimizing measurements is desirable from the standpoint of efficiency. The information plane must be able to provide information about paths between arbitrary end-hosts. This leads me to the underlying scientific question: what is the minimum set of measurements that the information plane needs to be able to provide this information?

1.2.4 Compatible with the growth of measurement vantage points: Scalable

The information plane's view of the network will only improve with the addition of more end-hosts from which it can perform measurements. This is because actual measurement of a path can be expected to be more accurate than estimation of the same. Further, some properties of the structure of the network local to an end-host may not be visible via measurements from other end-hosts. Given these benefits, the information plane should both incent more end-hosts to help in performing measurements and be able to scale accordingly.

End-hosts are likely to contribute measurements to the information plane only if they have an incentive to do so. The information plane can provide this incentive by ensuring that any end-host will be able to obtain significantly more accurate information about paths from it in return from measurements. The fact that the information plane will be able to do so is not readily apparent because even if an end-host is capable of providing measurements, measuring paths from that end-host to the rest of the Internet may not always be feasible, e.g., if the end-host has dialup Internet connectivity.

In order for the information plane to scale with the addition of measurement hosts, the measurement traffic sourced from each end-host should not significantly increase when more end-hosts actively participate. This would not only help the information plane sustain the increased load but is also desired from the perspective of efficiency.

1.2.5 Summary

In summary, the above set of challenges translate into the following goals in building an information plane: Internet-wide coverage, topology-awareness, accuracy, efficiency, and scalability.

1.3 Thesis and Contributions

In this dissertation, I support the following thesis: *it is practical to build a scalable system that measures the Internet from end-hosts and synthesizes these using a topology-aware methodology to estimate the latency and loss rate between arbitrary end-hosts on the Internet, with accuracy sufficient to improve the performance of distributed applications.*

The work presented in this dissertation makes the following contributions:

Algorithms for predicting the route through the Internet between arbitrary end-hosts. I develop algorithms that predict the route through the Internet between any arbitrary pair of end-hosts given a few measurements of the Internet's routing topology from the source and/or from the destination. I present and evaluate algorithms for doing so in two scenarios—first, when the input is a set of Internet routes, and second, when the input is a set of links that capture the Internet's structure. The two scenarios present a trade-off in accuracy versus the size of input. My algorithms for predicting routes do so at the right granularity required by the information plane—at the granularity that captures Internet performance, clustering together portions of the Internet are similar from a routing and performance perspective.

Scalable techniques for measuring the properties of Internet links. I develop measurement techniques for estimating the latencies of links in the Internet. Like existing Internet measurement tools, these techniques exploit properties of protocols in the Internet in novel ways. I also construct a simple algorithm that distributes the measurement load across end-hosts from which the information plane can issue measurements.

Design and implementation of iPlane, an Internet-wide information plane. I construct an information plane, iPlane, that is usable today by applications. iPlane constantly issues measurements from over 200 PlanetLab [65] nodes and over 500 public traceroute servers [92] to maintain a daily updated snapshot of the Internet's structure annotated with link metrics. iPlane exports an interface that end-hosts can use to contribute measurements and an interface that applications can use to query for information about path properties.

Demonstration of iPlane's benefit to distributed applications. My evaluation of the benefit of information provided by iPlane uses three representative applications—content distribution, peer-to-peer filesharing, and voice-over-IP. For each of these applications, I demonstrate that application performance significantly improves when information about the network is available from iPlane.

1.4 Organization

The remaining chapters of this dissertation are organized as follows. In Chapter 2, I provide an overview of the terminology used in this dissertation that is specific to literature related to the Internet, and I use this background to motivate my work. I present prior work in the areas of Internet measurement and overlay systems, areas of work that I build upon, in Chapter 3. In Chapter 4, I present my approach to predicting paths between arbitrary end-hosts by composing observed paths through the Internet. In Chapter 5, I then refine the path prediction algorithm to develop a model restricted to information on links seen across all observed paths. The focus of Chapter 6 is to outline the techniques I use to measure the latencies and loss rates of Internet links. I describe the Internet-wide information plane I built using the techniques presented in prior chapters and evaluate its utility in improving application performance in Chapter 7. I conclude in Chapter 8, summarizing the contributions of this dissertation and outlining the main areas of future work that can improve and build upon the work presented herein.

Chapter 2
BACKGROUND AND MOTIVATION

In this chapter, I introduce the Internet-related terminology that I use in the remaining chapters for describing the work presented in this dissertation. I breakdown the introduction of terms into three sections—describing the Internet's structure, the mechanisms by which routing works, and how both of these, together with traffic workload, determine path properties. I conclude by presenting the motivation for the problem I tackle in this dissertation and arguing for why this problem is hard.

2.1 Internet Topology

The basic goal of the Internet is to provide connectivity between end-hosts. Any end-host connected to the Internet should be able to communicate with any other end-host, except when communication with a particular end-host is actively filtered as a matter of policy. This global connectivity between end-hosts is enabled by networks administered by several thousand organizations, called Autonomous Systems (ASes).

Figure 2.1 shows the high-level overview of the Internet's structure. End-hosts are connected to ASes at the Internet's edge. They do so by means of access links, which are typically of much lesser bandwidth capacity than links that are within an AS (intra-AS links) or are between ASes (inter-AS links). The Internet is designed to make it efficient to statistically multiplex many users on to a smaller number of core links and routers. While ASes at the edge connect end-hosts to the rest of the Internet, ASes that are in the Internet's core connect other ASes with each other. Examples of ASes at the edge are Comcast and Verizon, whereas AT&T and Level3 are in the core.

The structure of a typical AS is shown in Figure 2.2. Every AS comprises routers and links connecting routers. A router has multiple router interfaces and a link connects a pair of router interfaces on two different routers. The function of a router is to forward a

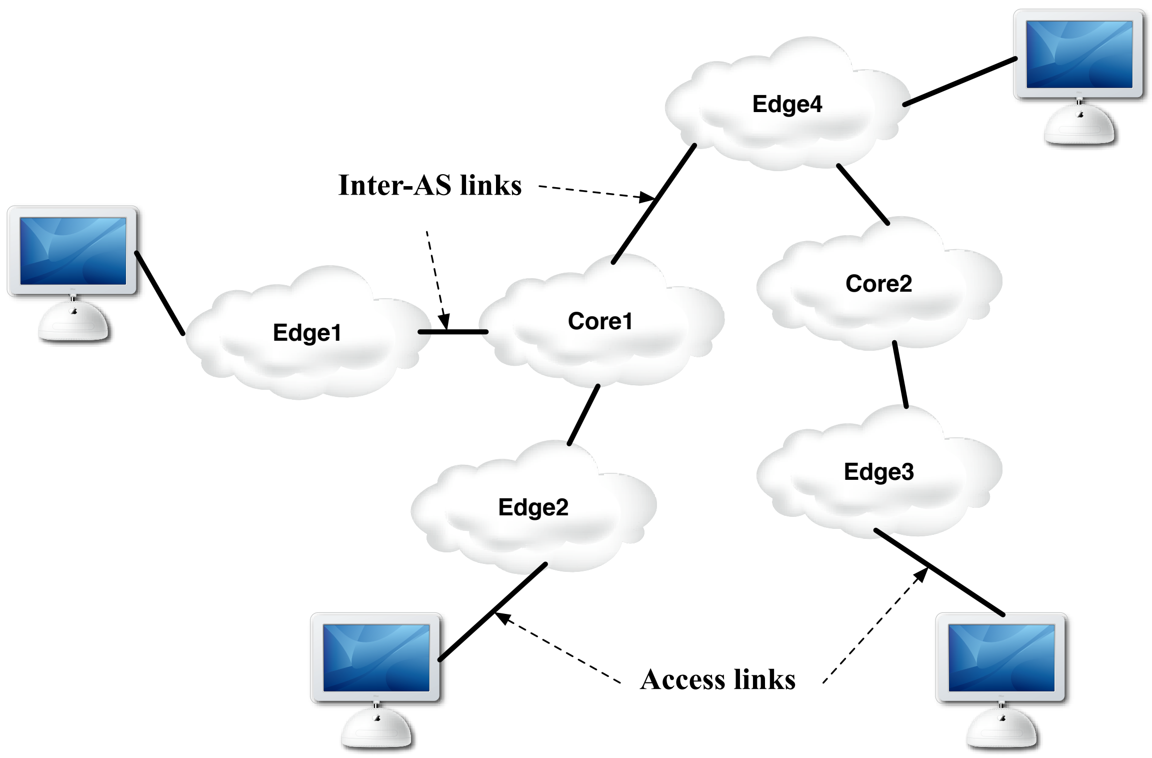


Figure 2.1: The Internet's structure comprises a graph of Autonomous Systems (ASes). End-hosts connect to ASes at the edge via access links, and ASes in the core provide connectivity between ASes.

packet it receives on one of its interfaces out on to the link connected with one of its other interfaces. All the interfaces on a router are said to be aliases of each other. Routers could be spread across several locations within an AS. Each location in which an AS has routers constitutes a Point of Presence (PoP) of the AS.

The Internet's connectivity is extremely rich. There are often many AS paths between two end-hosts. Two neighboring ASes can connect with each other at multiple PoPs, thus resulting in multiple possible PoP-level paths corresponding to the same AS path. Even within an AS, several paths typically exist between any two PoPs [88]. Moreover, end-hosts can opt to multi-home [2]—connect to multiple edge ASes. As a result of this rich connectivity, almost always, several paths exist connecting any two end-hosts. This had led

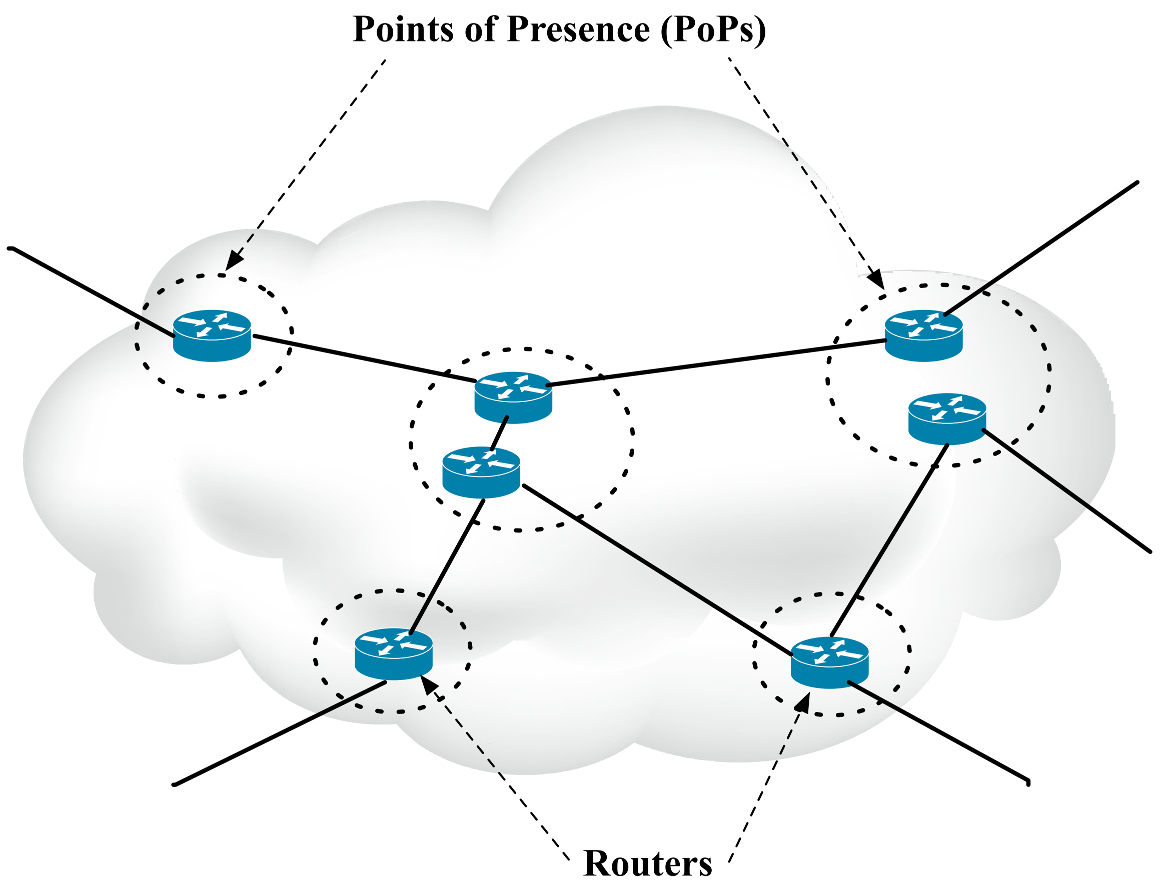


Figure 2.2: An AS can have routers in multiple locations. Every location in which the AS has routers constitutes a Point of Presence (PoP) of the AS.

to proposals to modify the Internet so as to enable either end-hosts [100] or the network itself [98] to take advantage of this redundancy.

2.2 Routing in the Internet

Routing serves the role of determining which of the myriad paths available between any pair of end-hosts should be used for communication between them. A router talks to its neighbors to determine which path to use for each packet; routing in the Internet is largely a function of the destination, though there has been some recent evidence [4] to the contrary. To setup the next-hop to every destination in each router, routing protocols are run in the

Internet at two-levels—inter-AS and intra-AS.

As shown in Figure 2.1, traffic between a pair of end-hosts may have to traverse multiple ASes. The routing protocol that ASes use to talk to each other, Border Gateway Protocol (BGP), announces the existence of paths to a particular destination only at the level of ASes. This is primarily due to two reasons. First, propagation of routes at the granularity of ASes ensures the Internet's scalability by isolating local routing decisions from affecting the global network. Second, since each AS is under a different administrative domain with ASes competing with each other for business, ASes do not want to reveal much about the structure of their network.

A simplified overview of how BGP works is as follows. For every destination, each AS receives an AS path from each of its neighbors. Based on a combination of these AS paths and the AS's policy-based decisions, the AS picks one of its neighbors as the next hop AS towards the destination. The AS then prepends itself to the AS path it had received from the chosen neighbor, and announces this modified AS path to some subset, possibly all, of its neighbors. An AS filters propagation of routes to its neighbors based on its export policy. In the absence of any changes to the Internet, this process would eventually converge resulting in a chosen AS path from every AS to every destination in the Internet.

However, an AS path alone does not precisely determine how the packets sent from a source end-host will be forwarded on to the intended destination. Routers are the entities forwarding packets, and hence, the route from a source to a destination needs to be determined at router-level. The router-level path corresponding to an AS path is made ambiguous by the facts that neighboring ASes can connect with each other at multiple locations and even within an AS, there can be multiple paths between a pair of routers. Therefore, every AS needs to run a routing protocol within its network to determine the route from each of its routers to the next hop AS for every destination.

The goal of routing protocols that run within an AS, referred to as Interior Gateway Protocols (IGPs), is to setup a forwarding table within each router. This forwarding table contains, for every destination, the address of the next hop router. The forwarding tables are typically computed so as to either minimize the latency experienced by packets while traversing the ISP or to load balance the expected traffic across all links in the ISP's network.

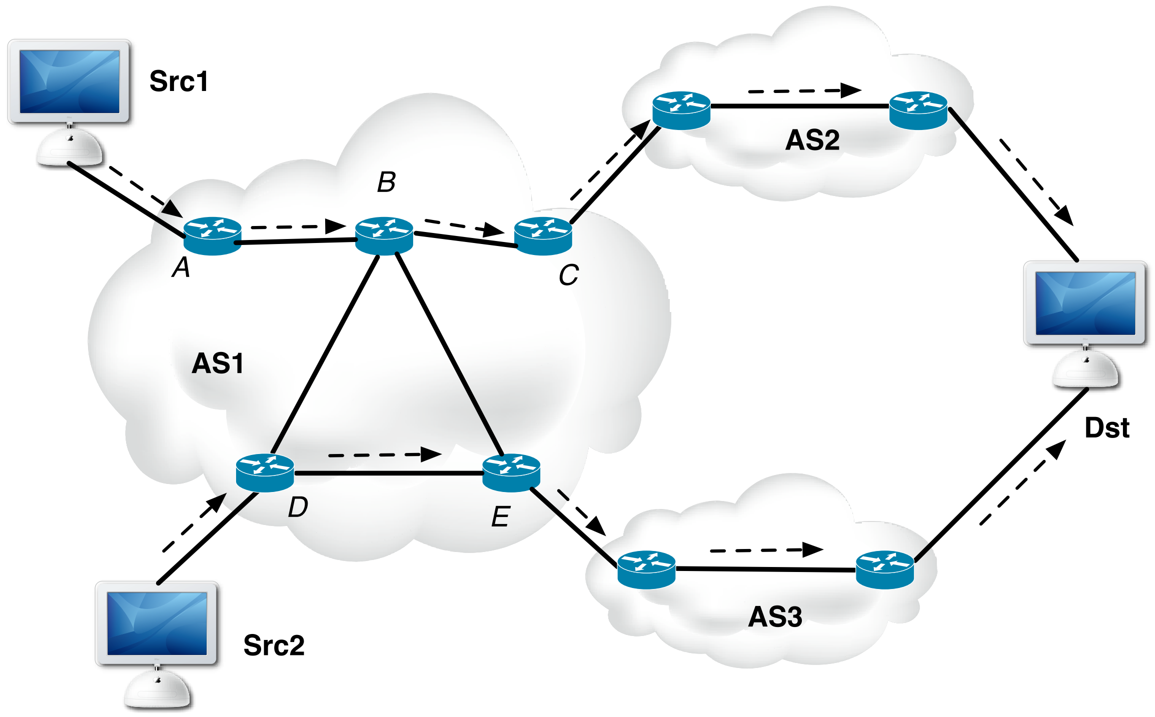


Figure 2.3: Routes chosen from *Src1* and *Src2* to *Dst* when *AS1* performs early-exit routing. Solid lines show links between routers, and dashed lines indicate the links along which packets destined to *Dst* are forwarded. The AS path *"AS1 AS2"* is used for traffic from *Src1* to *Dst* because the ingress router *A* is closer to the egress router *C*. Whereas, the AS path *"AS1 AS3"* is used between *Src2* and *Dst* because the ingress router *D* is closer to the egress router *E*.

Examples of IGPs are OSPF and IS-IS.

Note that the outcomes of BGP and IGP are tightly linked to each other. It is not the case that an AS uses BGP to choose the AS path to a destination and then determines the router-level path using IGP. To the same destination, an AS can choose to forward traffic along different AS paths from different parts of its network. For example, many ASes choose to implement early-exit routing [89, 80]. In early-exit routing, every router in the AS chooses to forward traffic towards the closest edge router of the AS from where an AS path to the destination is available. An AS can be connected to different neighboring ASes at different edge routers. Therefore, as a result, all routers within an AS need not necessarily forward traffic to a destination to the same next AS. Figure 2.3 shows an illustrative example of how early-exit routing can lead to different AS paths from different sources to the same destination.

Internet routes can also be asymmetric; the route, both at the granularity of ASes as well as routers, from a source to a destination need not necessarily be the reverse of the route back from the destination to the source. This is because each AS has its own routing policies, and it chooses the next AS or router towards any destination independent of the choices made by other ASes. So, for example, in Figure 2.3, even though *AS1* forwards traffic from *Src1* to *AS2* on the way to *Dst*, *AS2* might choose to forward the traffic back from *Dst* via some AS other than *AS1*. Paxson found more than half the routes he measured to be asymmetric [63].

Note that I have so far presented routes as being per-destination to simplify the discussion. In reality, end-hosts are grouped into address blocks called IP prefixes. A prefix comprises all IP addresses with p as their first n bits typically written as p/n. Routers maintain routes per-prefix. When a router receives a packet destined to a particular destination, it determines the most selective prefix that this destination belongs to, i.e., among the p/n's that the destination address matches, the one with maximum value of n. The router then forwards the packet along the route associated with this prefix.

Further, prefixes can be grouped together into BGP atoms [7]. Though routers compute routes at the granularity of prefixes, it has been observed that prefixes can be clustered into groups such that the AS path to each of the prefixes in a group is the same from any given vantage point. Such groups of prefixes are called atoms.

2.3 Path Properties

Application traffic workload on the Internet is constantly in flux. As a result, the amount of bandwidth consumed on each link in the Internet varies over time. ISPs monitor the usage of links in their network and tune their intra-AS and inter-AS routing policies to ensure any part of their network does not get congested. Routes through the Internet can change as ISPs tweak their routing policies under the constraints imposed by the structure of their networks. Hence, the properties of the path between any pair of end-hosts—determined by the route between them and properties of the links on the route—depends on a complex combination of structure, routing, and traffic workload of the Internet.

I focus mainly on two path properties in this dissertation—latency and loss rate.

The latency between a source and a destination is defined as the time taken by a packet sent from the source to reach the destination and for the response to then come back to the source. I only consider the latency experienced by packets in traversing the Internet. Though the latency seen from an application's perspective may depend on other factors such as processing load on end-hosts, which can cause packets to be queued in the kernel on the source or the destination, such factors are outside the scope of my work.

Latency through the Internet has several components. First, a packet undergoes *transmission delay* on each link along the path. This is the time taken to send the packet from a router's queue out on to one of its links. Transmission delay is computed as *(size of the packet) / (bandwidth of the link)*. Second, the packet is subjected to *propagation delay* on each link—the time taken for the packet to travel from one end of the link to the other end. Propagation delay has a lower bound of *(distance covered by the link) / (speed of light in the physical medium used by the link)*. Third, the *queueing delay* on packets as they wait in packet buffers on routers adds to the latency. Lastly, each router along the path also adds a store-and-forward delay, which is usually minimal. In this dissertation, I restrict latency to be the sum of propagation and queueing delays. My assumption of ignoring transmission delay is largely fine since the ubiquitous deployment of high bandwidth links ensures that transmission delay is insignificant compared to the other components of latency, though this does not necessarily hold true for access links [20].

Loss rate along a path is the probability that a packet sent from the source will not reach the destination. Packets may be dropped in between due to numerous reasons. The failure of a router or link can result in there being no path between the source and destination. Even after failed routers/links are repaired or if an alternate path is available, it can take a while for routing to converge [43]. In the meanwhile, packets can be dropped due to the absence of a route to the destination. Further, even when a route exists, some packets might be dropped enroute due to lack of space in some router's packet buffer or due to transmission error, e.g., in wireless networks. In this dissertation, I focus on measuring and estimating the loss rate caused by this last factor—loss that occurs on a path because of competition with traffic of other applications using routers/links along the path.

2.4 Need for an Information Plane

Applications would prefer to use paths with low latencies and low loss rates for communication. Lower latency implies faster exchange of packets between source and destination. Lower loss rate reduces the need to retransmit packets and the timeouts necessary to recognize the need for retransmissions. However, the Internet currently does not provide latency and loss rate information to applications. Instead, applications themselves need to measure the properties of paths, and choose paths with better properties for communication. As I argued previously in Chapter 1, this status quo is inefficient and also onerous on applications.

The goal of my work in this dissertation is to build a common information plane for the whole Internet that can provide latency and loss rate information to applications. My approach for estimating the path properties between a pair of end-hosts is two-stage—predict the route between them and then compose the properties of links on the predicted route. End-to-end latency is a summation of link latencies. Assuming packet loss on different links occurs independently, end-to-end loss rate can be computed using link loss rates based on the property that the probability of a packet not being dropped along a path is equal to the probability the packet is not dropped on any of the links on the path.

Both predicting routes and measuring link properties are hard. There are multiple paths to choose from between any pair of end-hosts and which one of these is chosen depends on the undisclosed routing policies of several ISPs. Moreover, routes are not constant; they can change due to changes in the Internet's structure, router failures, or changes in traffic workload. These causes for flux in the Internet also make it hard to measure link properties. Link properties need to be measured often enough to keep up with these changes. Measurement of link latencies is made challenging because of path asymmetry. In the remaining chapters of this dissertation, I describe how I tackle these challenges to build an Internet-wide information plane.

Chapter 3
RELATED WORK

In this chapter, I present the existing work that this dissertation builds upon. My work leverages work in two broad areas—measuring the Internet and building overlay systems—and I outline the related work in these areas. I also describe prior work towards building an Internet-wide information plane and distinguish the contributions of my work from these.

3.1 Internet Measurement

The Internet provides best-effort forwarding of packets. It does not provide any guarantees on its performance nor does it support querying for the performance it can currently provide. Therefore, measuring the Internet is important from the perspective of both applications and ISPs. Large-scale distributed applications, which have many potential paths to choose from for any given communication, need to measure the network to distinguish good paths from bad paths. ISPs need measurements both for accounting as well as for monitoring their networks. Measurements of the Internet are also necessary to drive the design of the next-generation Internet and its applications. Given the various benefits of Internet measurement, this topic has received a lot of attention in the research community over the past decade.

Research in the area of Internet measurement can be classified into three categories—devising new techniques and building new tools to measure the network, using these to perform large-scale studies that characterize the Internet's properties, and building systems that use measurements to monitor the Internet.

3.1.1 Measurement tools and techniques

The Internet is not designed to provide support for measurement. The Internet's control plane, which includes protocols such as SNMP, only enables ISPs to monitor their own networks. However, the networking community has managed to leverage features of Internet protocols in novel ways to develop tools and techniques that enable end-hosts to measure various properties of paths or even individual links.

`ping` [55] and `traceroute` [38] are the basic tools for Internet measurement. ping sends several packets as probes to a specified destination and reports 1) the number of probes for which responses were received, and 2) the latency from the probing source to the destination. Thus, ping can be used both to measure latency and loss rate of path. If ping is used to send a large number of probes and if it does not receive responses for any of the probes it sends, it implies that the destination is unreachable or unavailable for communication.

Traceroute measures the router-level path to a destination. The Internet Protocol (IP) has a Time-To-Live (TTL) field that is supposed to be decremented by every router when it forwards a packet. When a router receives a packet with a TTL value of 1, it drops the packet and returns a response indicating an error to the source of the packet. The original intent of the TTL field was to ensure packets do not oscillate inside the network forever. Instead, traceroute utilizes the fact that sending out a packet with TTL set to k will elicit either a response from the k^{th} router on the path to the destination or a response from the destination if it is within k hops away. The standard implementation of traceroute sends three probes for each TTL value to tolerate packet losses. It starts off with a TTL value of 1 and increments the TTL until either the destination is reached or an error response is received that the destination is unreachable.

Figure 3.1 shows the output of a sample execution of traceroute. Each pair of successive lines in the output provides two pieces of information. For example, consider the lines corresponding to hops 10 and 11 in Figure 3.1. First, discounting for load-balancing [4] and route changes during the traceroute, this indicates the presence of a link between the routers with addresses *69.28.172.41* and *206.223.119.104*. Second, the output also yields the information that on receipt of packets destined to the address *194.102.253.138*, the router *69.28.172.41* forwards the packets to *206.223.119.104*. Thus, traceroute discovers both structure and routing information.

Ping and traceroute help measure the latency and route to any destination. Similarly, many other properties of the network can be determined by *actively* sending out coordinated

```
traceroute to usis.kappa.ro (194.102.253.138), 64 hops max, 40 byte packets
 1 acar-atg-02-vlan76.cac.washington.edu (128.208.3.102) 8 ms 1 ms 3 ms
 2 uwcr-ads-01-vlan3805.cac.washington.edu (205.175.108.21) 0 ms 0 ms 0 ms
 3 uwcr-ads-01-vlan1839.cac.washington.edu (205.175.101.157) 0 ms 0 ms 0 ms
 4 uwbr-ads-01-vlan1802.cac.washington.edu (205.175.101.10) 0 ms 0 ms 0 ms
 5 iccr-sttlwa01-02.infra.pnw-gigapop.net (209.124.188.132) 0 ms 0 ms 0 ms
 6 pnwgp-cust.tr01-sttlwa01.transitrail.net (137.164.131.186) 0 ms 0 ms 0 ms
 7 te4-3--301.tr01-sttlwa01.transitrail.net (137.164.131.185) 0 ms 0 ms 0 ms
 8 137.164.129.3 (137.164.129.3) 51 ms 51 ms 51 ms
 9 llnw-peer.chcgil01.transitrail.net (137.164.130.166) 61 ms 51 ms 51 ms
10 ve6.fr3.ord.llnw.net (69.28.172.41) 61 ms 51 ms 61 ms
11 us-chi01a-ri1-ge-0-0-0.aorta.net (206.223.119.104) 61 ms 61 ms 61 ms
12 213.46.190.89 (213.46.190.89) 81 ms 81 ms 81 ms
13 uk-lon02a-rd1-so-4-0-0.aorta.net (213.46.160.205) 153 ms 153 ms 153 ms
14 uk-lon01a-rd3-ge-2-0.aorta.net (213.46.174.33) 157 ms 157 ms 156 ms
15 de-fra01a-rd1-pos-2-0.aorta.net (213.46.160.9) 170 ms 170 ms 170 ms
16 de-fra01a-rd2-ge-3-1.aorta.net (213.46.160.58) 192 ms 192 ms 192 ms
17 ro-cj01a-rc1-10ge-2-1.astralnet.ro (213.46.170.70) 192 ms 192 ms 192 ms
18 ro-buh01a-rd1-v800.astralnet.ro (193.230.240.150) 281 ms 213 ms 213 ms
19 xcr1-BR-175.b.astralnet.ro (194.102.255.65) 206 ms 206 ms 206 ms
20 ns.usembassy.ro (194.102.253.138) 214 ms 216 ms 219 ms
```

Figure 3.1: Output of `traceroute` from the University of Washington to the US embassy in Romania. Each line corresponds to one hop with the DNS name and IP address of the router at the hop, and the latencies experienced by the three probes to that hop.

packets as probes into the network and monitoring the responses received. Tools that perform active measurements have been built to measure several path properties, among which are loss rate, bandwidth capacity, and available bandwidth.

Tulip [49] attempts to measure the loss rate on links on the path to any destination. It can also detect instances when the packet probes it sends out to measure loss are either queued or reordered. Tulip measures the loss rate resulting from packet loss on the round-trip path to the destination and back, and it uses periodic ICMP or UDP packets as probes to measure the aggregate loss rate over a period of time. In contrast, Sting [72] uses TCP probes to distinguish between loss on the forward and reverse paths, and Badabing [78] issues probes at a non-uniform rate to detect packet loss episodes.

Tomography infers the presence of a common link between paths from a source to two destinations by observing that any change in the properties on the path to one of the destinations is reflected on the path to the other. Padmanabhan *et al.* [62] used this approach to measure link loss rates by passively monitoring traffic at a webserver. The MINC project [66, 8] used multicast probes to measure link latencies and loss rates. They later extended [22] their work to infer loss rates using striped unicast probes—probes sent out to different destination with no inter-probe spacing. My approach to measuring link properties is more direct than network tomography. I measure the presence of a link using traceroutes and then measure the link's properties such as latency and loss rate.

Passive monitoring of real traffic can also help measure several properties of the network. `TRAT` [104] monitors TCP traffic to determine RTTs to destinations contacted. Passive inspection of traffic received at a webserver was used to determine link loss rates [62]. PlanetSeer [103] detected failures in the network by passively monitoring traffic received by the CoDeeN [94] content distribution network. Recently, Casado *et al.* [11] have also proposed the use of spurious traffic such as worms and spam for passive measurement of the Internet.

My focus in building an information plane is to leverage this large body of existing work for measuring a diverse set of path properties, rather than develop new measurement tools or techniques. As the Internet measurement community improves on existing measurement techniques, the information plane can keep abreast by using the most accurate and efficient tools available.

A couple of key techniques I leverage from prior work are alias resolution—discovering which router interfaces correspond to the same router—and the clustering of routers into PoPs. To discover candidate router interface pairs that could be aliases, I reuse two existing techniques. First, I use the insight from Mercator [30] that UDP probes to interfaces on the same router elicit responses with the same source address. Second, I discover interfaces potentially on the same router using the fact that interfaces on either end of a link are usually in the same */30* prefix [102]. I then use a combination of known alias resolution techniques [79] to determine which of the alias candidates are indeed on the same router. To cluster routers into PoPs, I use Rocketfuel's undns [81] utility to map DNS names of interfaces to geographic locations. Using all of the above known techniques in the literature, in combination with a few modifications of my own, I generate a coarse-grained map of the Internet's structure annotated with routing policies and properties of links and path segments.

3.1.2 Measurement studies

Empowered by ping, traceroute, and other measurement tools, there have been several efforts to measure the network at scale. Paxson's work [63] on monitoring routes between nodes on the NIMI testbed was one of the seminal pieces of work on Internet measurement. Paxson repeatedly performed traceroutes between 37 NIMI sites over a couple of two-month long periods, once in 1994 and once in 1995, to study various characteristics of routing—routing pathologies, stability of routes, and routing asymmetry. His main findings included: 1) the likelihood of a major routing pathology increased from 1.5% to 3.3% from 1994 to 1995, 2) two-thirds of routes were stationary across days and even weeks, and 3) at least half the routes were asymmetric. Though these numbers might have changed over the past decade with the rapid growth of the Internet, the existence of significant routing stability and routing asymmetry provides some guidance in building an information plane.

Rocketfuel [81] orchestrated measurements from several hundred public traceroute servers to discover the physical topology of many ISPs. The primary challenges that Rocketfuel tackled in discovering maps of ISP structures were accuracy, i.e., processing traceroutes to detect the correct nodes and links, given that routes can change during traceroute, and efficiency, i.e., minimize the number of measurements necessary to infer the correct topology. Mercator [30] had similar objectives but issued measurements from a single vantage point. My work extends beyond Rocketfuel's and Mercator's goals in three directions: 1) I strive to measure the PoP-level structure of the whole Internet, and not just that of a few specific ISPs, 2) I go beyond structure and measure the properties of every link, and 3) the information plane I build, iPlane, refreshes the measured structure annotated with link properties continually.

There have been numerous other measurement studies—too many to be comprehensively listed here. I mention here the few that are either closely related to this dissertation, or whose insights I use in this work. Zhang *et al.* [105] studied the stationarity of routing and of path properties such as loss rate and TCP throughput, and Rexford *et al.* [69] showed that BGP paths to popular destinations are remarkably stable compared to overall BGP stability. Donnet *et al.* [21] demonstrated the significant redundancy in measurements when traceroutes are performed from several vantage points to a large number of destinations. They showed that routes near the sources and near the destinations are redundantly probed several times over and proposed reducing this redundancy by staging the TTL ranges for each traceroute.

My work leverages the insights gained by the measurement community in these and other efforts to build an information plane that continually monitors paths and path properties on the Internet.

3.1.3 Measurement systems

Apart from measuring the Internet to study its properties, several systems have also been built that continually measure and monitor either portions of the Internet or networks internal to a particular organization. Several efforts have looked at building information planes that monitor end-host performance and at optimizing the query processing engine of the information plane. Examples include Sophia [95], PIER [35], IrisNet [29], SWORD [60], HP's OpenView [59], and IBM's Tivoli [91]. The Scalable Sensing Service (S^3) [44, 74] monitors the properties of paths between PlanetLab nodes. S^3 measures the latency, loss rate, bandwidth capacity, and available bandwidth between every pair of PlanetLab sites once every four hours.

All of these systems have a different focus than mine. They manage information about nodes (e.g., PlanetLab nodes, routers in an ISP, or sensors), and paths between these nodes, under control of the system. In contrast, I target estimation of path performance at Internet-scale. Hence, I study the prediction of properties along paths where I either do not have control over the source, or over the destination, or possibly both. While the vision of how to go about building an Internet-wide information plane has been discussed before [15, 84], I present a realization of this vision in the form of iPlane.

Like the information plane I present in this dissertation, there have been several other Internet-wide measurement systems built recently,but with objectives different from mine. Hubble [41] monitors reachability of all Internet hosts to detect instances when packets do not reach a destination even when a BGP path to it exists. Hubble attempts to diagnose and classify the cause of reachability problems that it detects. Zhang *et al.* [106] use collaborative probing from a set of end-hosts to diagnose routing disruptions in specific ISPs. Netdiff [50] measures path performance from backbone ISPs to the rest of the Internet so that this information can be used by other ISPs in making decisions related to peering.

Several systems have also been built with the sole goal of maintaining an up-to-date map of Internet routes. RouteViews [53] constantly gathers BGP routing updates from several ISPs around the world providing a current map of the Internet's AS-level routing topology. Skitter [9] continually performed traceroutes to destinations in all routable prefixes and made these traceroutes available for other researchers. DIMES [75] provides a measurement agent that users can download and run on their computers. The central DIMES node orchestrates pings and traceroutes from all agents, gathers these measurements, and then synthesizes them to construct maps of the Internet's topology. iPlane goes beyond explicit measurement and makes predictions of routes and path properties along unmeasured paths.

iPlane's approach of reasoning about the Internet after discovering its structure has also been used to improve the ability to identify the geographic location of hosts on the Internet. Initial work at mapping IP addresses to geographic locations relied solely on latency measurements from vantage points to end-hosts. Katz-Bassett *et al.* [40] and the Octant system [97] have shown that the hop-by-hop latency constraints introduced by discovering the Internet's routing topology help significantly improve the accuracy of IP geolocation. The focus of my work is different—to estimate path properties on the Internet.

3.2 Prior Work towards an Information Plane

In several measurement systems, some measurements will need to be inferred from others. This could be due to one of three reasons—1. to minimize the number of measurements the system performs, 2. the scale of the system may make it infeasible to perform all necessary measurements, or 3. measurement overhead may not be affordable at the time when the result of some measurement is needed.

The measurement inference problem has received a fair bit of attention recently, specifically with regard to latency. The objective is to infer latencies between all pairs of a set of end-hosts given latencies for a subset of these pairs. IDMaps [24] is an early example of a network information service that tackles this problem by using a small set of vantage points as landmarks. Subsequently, GNP [56] pioneered the approach of embedding end-hosts in a low-dimensional Euclidean space. Several following works such as NPS [57], Lighthouse [64], Lim *et al.* [45], Tang *et al.* [86], Vivaldi [18], and PIC [17] have built on top of this basic approach to provide decentralization [57, 18], improved computational efficiency [86], resilience to measurement error [17, 18], security [17], and accuracy. The techniques used to minimize error range from Simplex minimization [17, 57] to Principal Component Analysis (PCA) [45, 86] to spring relaxation [18] algorithms. Approaches to embed Internet distances into non-metric spaces, such as using hyperbolic coordinates [76] and using matrix factorization [51], have also been attempted. A popular metric used for demonstrating the accuracy of these approaches is the absolute relative error, which the study by Lua *et al.* [46] suggests does not necessarily indicate usefulness of the estimate for various applications.

I next explain a couple of these latency inference techniques in more detail—IDMaps [24], because it uses measurements from vantage points similar to iPlane, and Vivaldi, one of the best of the various coordinate-based approaches. IDMaps issues pings from a bunch of vantage points to all participating end-hosts and also measures latencies between all pairs of vantage points. To estimate the latency between a pair of end-hosts S and D, it first identifies the vantage points V_S and V_D with the lowest latencies to S and D, respectively. IDMaps then computes the end-to-end latency between S and D as the sum of latencies along paths (S, V_S), (V_S, V_D), and (V_D, D). This technique is based on the assumption that a vantage point can be found close to any end-host, which would make the path between the vantage points representative of the path between the source and destination. However, for this to work at Internet-scale, millions of end-hosts would be required, which would make the measurement of latencies between all pairs of vantage points infeasible. iPlane uses a similar approach of issuing measurements from a set of vantage points but replaces pings with traceroutes. First, beyond latency, iPlane also predicts the route between a pair of end-hosts S and D and the other path properties such as loss rate. Second, rather than relying on having vantage points with negligible latency to S and D, iPlane relies on having a vantage point with a route to D that's similar to the route from S.

Vivaldi [18] is a fully distributed coordinate system. Vivaldi models nodes as being at a certain height above a plane in order to reflect the impact of the first hop. The distance between two nodes is the sum of their heights plus the distance along the plane. Vivaldi adjusts coordinates to minimize error. Every pair of nodes in the system is modeled as connected by a spring. The spring connecting two nodes is in a stretched or compressed state corresponding to when the distance predicted between them by their coordinates over-estimates or under-estimates the actual latency. Each node computes its own coordinates to minimize the local error in its neighborhood. Vivaldi's modeling of the Internet delay space with an Euclidean metric leads to systemic inaccuracies because it cannot capture triangle inequalities, a commonly known occurrence [73, 3] in the Internet. iPlane gets around such inaccuracies by using a structural approach.

Systems that tackle the latency estimation problem in specific contexts such as determining the closest among a set of nodes to a particular end-host [26, 96], or determining the central leader among a set of nodes [96] have also been developed. Though most of the above techniques and systems are simple to implement, it is not apparent how any of them can be extended to other path properties such as loss rate or bandwidth capacity.

BRoute [33] is a system for available bandwidth estimation that scales linearly with the number of nodes. Chen *et al.* [13] and Zhao *et al.* [107] use an algebraic formulation to determine the minimum subset of paths that need to be measured to estimate loss rates. However, these systems require control over all end-hosts on paths for which properties are to be inferred. Since an Internet-wide information plane will not have access to measurements along the path between every pair of end-hosts, I develop a structural prediction technique that uses available measurements of the Internet's structure to predict the path between arbitrary end-hosts. I use this predicted path to estimate end-to-end latency and loss rate.

Inference of paths has also been studied, predominantly at the AS-level. Mao *et al.* [52] describe a structural inference approach called RouteScope to infer AS-level paths. They use constrained optimization to model aspects of interdomain policy routing such as valley-free routing and the preference of customer ASes over peers over providers, and use additional measurement techniques to observe routes from multihomed prefixes. Muhlbauer *et al.* [54] attempt to develop a hybrid model of Internet routing that lies in between a blackbox and a structure inference approach. They introduce "quasi-routers" to model the presence of multiple border routers in an AS based on an observed set of routes. Their approach can predict the training set exactly and achieves 50% prediction accuracy for unobserved routes. There has also been work towards inferring the intra-AS [48] and inter-AS [80] policies that drive routing. In this dissertation, I advance the state of the art in this domain by presenting algorithms that infer the PoP-level paths between arbitrary end-hosts on the Internet.

3.3 Overlay Systems

A potential approach towards the goal of my dissertation—providing information about path properties on the Internet to applications—is to modify the Internet's control plane such that the Internet itself continuously monitors its path properties and end-hosts can query for this information (a la knowledge plane [15]). However, any single ISP does not have an incentive to incorporate such a modification into its network until all other ISPs do so. The networking community's experience over the last decade has shown that research proposals rarely make it into operational networks in such situations. Hence, the approach

I take in building an information plane is to monitor the Internet by issuing measurements from end-hosts.

This approach is similar to prior work that implement new functionalities at the application layer even though these solutions theoretically could be incorporated into the Internet itself. Such systems that implement functionality over the network are called overlay systems. ESM [14] and Overcast [39] implement multicast by having end-hosts participating in the system serve as intermediate relays, rather than have routers relay traffic like in IP Multicast. The Detour [73] project demonstrated that the path performance between a pair of end-hosts on the Internet could be improved in some instances by detouring the traffic from the source to the destination via another end-host. RON [3] implemented the vision of Detour on a testbed of 30 or so nodes. Every node in the RON testbed continually monitors reachability and performance to every other RON node, and detours its traffic through another RON node when its measurements indicate that the detour path can yield benefits over the default Internet path. RON's necessity to constantly measure paths through all possible detour options limits its scalability. SOSR [31] addressed this limitation by demonstrating that most of the reliability benefits of RON can be obtained by instead detouring traffic through four randomly chosen detour nodes. OverQos [85] enhanced the Internet's quality of service by routing traffic through an overlay network. Instead of requiring the reservation of resources on links in the Internet, OverQos routes traffic through paths that have predictably good properties.

While all of these overlay systems attempt to provide a layer over the Internet on top of which applications can be built, my work is targeted at large-scale distribution applications built over the Internet. These applications are overlays themselves, since they run on end-hosts and their communication patterns define a logical network on top of the Internet. iPlane is intended to provide information about Internet paths to help applications setup edges in their overlay network such that application performance is optimized.

Chapter 4
PATH PREDICTION BY COMPOSING PATHS

In this chapter, I seek to develop a model that uses measurements of the Internet's structure and routing topology to predict the route between any pair of Internet hosts. In subsequent chapters, I present how to use this structural model of the Internet to estimate path properties—data that the information plane needs to provide.

4.1 Overview

In pursuing a structural approach towards building an information plane, I implement the following methodology. To estimate the path properties between any arbitrary pair of end-hosts, I first predict the routes between these end-hosts, both in the forward and reverse directions. I separately predict the path in the forward and reverse directions to reflect the prevalence of asymmetric routing in the Internet [63]. I then compose the inferred properties of either links or path segments along the predicted forward and reverse routes to derive an estimate for the end-to-end path property.

In this chapter, I address the first piece necessary to implement the above methodology—predicting routes between end-hosts. The technique that I employ to predict routes between a pair of nodes is to compose partial segments of known Internet paths. I refer to this technique hereafter as the *path composition* technique. The paths used in path composition are obtained from an *atlas* of the Internet that is measured and maintained by a set of geographically distributed vantage points—end-hosts which can perform measurements. The atlas comprises the topology of the Internet core and some selected portions of the edge, and it also includes round-trip latency measurements from the vantage points to the mapped portions. When an end-host joins the infrastructure, the atlas is augmented with a few paths from the joining node. The entire set of paths is then used during path prediction.

The basic principle underlying the path composition approach is to exploit similarity of routes. Since the Internet predominantly uses destination based routing, routes from sources that are close by will tend to converge when heading to the same destination. For example, when both sources are in the same AS, early convergence of routes will occur in the case of both early exit (both take the same nearest exit) and late exit (both take the exit nearest the destination). So, my hypothesis is that, given a sufficient number of geographically distributed vantage points, the route from any source to a destination will have a significant overlap with some path segment from one of the vantage points. I make accurate route predictions by maximizing this overlap, subject to the optimization of key routing metrics, such as AS path length, hop counts, and latency values.

Gathering a comprehensive atlas of the Internet from a limited number of vantage points is however infeasible given the Internet's size. Therefore, the path composition technique has to work with an incomplete view of the Internet's topology. In fact, my evaluation will show that a comprehensive router-level atlas of the Internet is not required to make accurate predictions of path performance. I operate at the right granularity to capture path performance by clustering together portions of the Internet similar from a routing and performance perspective. I do this by by first grouping together multiple router interfaces on the same router and then identifying all routers in the same Point of Presence (PoP) in an AS.

The path composition technique however can yield several candidate paths between a pair of end-hosts. This is a result of the rich connectivity of the Internet; there do exist several physical paths between a pair of hosts. One among the set of paths is chosen by the Internet's routing protocols based on the routing policies enforced by ASes. Therefore, I attempt to infer routing policy information from the set of observed routes gathered to build the Internet atlas. I use this routing policy information to distinguish amongst the different potential paths returned by the path composition technique.

In this chapter, I first present my goals in developing a path prediction technique and the sources of measurements I use. I then describe each of the three pieces that make path prediction work—path composition, clustering, and inference of routing policy. A summary of the various techniques presented in this chapter is listed in Table 4.1.

Table 4.1: Summary of techniques employed in constructing an atlas of Internet routes and using it to predict paths between end-hosts.

Technique	**Description**	**Goal**	**Section**
Traceroutes from vantage points	Paths to all prefixes/atoms are measured from a large number of geographically distributed vantage points	Build router-level atlas	Section 4.3
BGP snapshot from RouteViews	The BGP snapshot available from RouteViews is used to infer the origin AS of each router interface and to partition all routable prefixes into BGP atoms	Partition routers based on policy and routing	Section 4.3 and 4.5
Path composition	Observed routing segments from a source and to a destination are composed to predict a path between the source and destination	Predict end-to-end paths	Section 4.4
Alias resolution	Determine if router interfaces are on the same router	Cluster router interfaces into routers	Section 4.5
Location mapping	Map router interfaces to locations based on DNS names	Cluster routers into PoPs	Section 4.5
Return TTL clustering	Router interfaces that return similar TTLs to a large number of vantage points are clustered together	Cluster routers into PoPs	Section 4.5
Shortest AS path and early-exit	Choose among path segments that can be composed	Model default routing policy	Section 4.6.1
AS three-tuples	Gather all sequences of three successive ASes seen in observed paths	Capture path export policies	Section 4.6.2

4.2 Goals

I seek to develop a technique for predicting paths with the following three main goals in mind.

- **Accuracy:** My primary goal is to make accurate predictions of routes. In my structural approach, the path properties between a pair of end-hosts are computed by composing the properties of links and path segments along the predicted route between the hosts. Therefore, getting route prediction right is critical for my whole methodology to work.

- **Efficiency:** I seek to predict routes while minimizing the number of measurements required as input. My aim is to get at the underlying scientific question: what is the minimum set of measurements required to make Internet-wide prediction of routes?

- **Scalability:** Path prediction needs to work using measurements predominantly from a limited number of vantage points with only a few measurements from every end-host. This is necessary not only to ensure that I can build an Internet-wide information plane usable by applications today without waiting for worldwide deployment but also to ensure the information plane is scalable. If the information plane were to rely on a comprehensive set of measurements from every end-host, the system would not scale to the whole Internet.

4.3 Building an Atlas

The path composition approach works by splicing together segments of observed paths to predict routes along unmeasured paths. Hence, I accumulate an atlas of Internet routes. I gather route information from two sources—1) traceroutes from vantage points and 2) AS paths from BGP feeds.

My primary tool for building the atlas is `traceroute`, which allows me to identify the network interfaces on the forward path from the probing entity to the destination.

`traceroute` also provides me with hop-by-hop round-trip times, which I use to model some aspects of routing policy, e.g., early-exit routing.

I need geographically distributed vantage points in order to map the Internet topology and obtain a collection of observed paths. PlanetLab servers, located at over 300 sites around the world, serve as the primary vantage points. Probes can be issued from PlanetLab nodes at a reasonably fast rate; a traceroute probe every second translates to measurement traffic of about 4KB/s, a modest load on the PlanetLab nodes. I also enlist the use of around 500 public Looking Glass/Traceroute servers for low-intensity probing. I issue probes from these vantage points to representative IP addresses in routable prefixes, sometimes choosing just one IP address for a collection of prefixes, thereby minimizing the measurement load.

BGP snapshots, such as those collected by RouteViews [53], are a good source of probe targets. To achieve a wide coverage for the topology mapping process, I obtain the list of all globally routable prefixes in BGP snapshots and choose within each prefix a target *.1* address that responds to either ICMP or UDP probes. A *.1* address is typically a router and is hence more likely to respond to probes than arbitrary end-hosts. Of all of the targets discovered in a prefix, one is chosen at random as the representative end-host for that prefix.

Measurement load can be reduced further by clustering IP prefixes into BGP atoms [7] for generating the target list. A BGP atom is a set of prefixes, each of which has the same AS path to it from any given vantage point. BGP atoms can be regarded as representing the knee of the curve with respect to measurement efficiency—probing within an atom might find new routes, but it is less likely to do so [7]. However, in this dissertation, I consider the atlas to comprise traceroutes to one target in every prefix.

I use the PlanetLab nodes to perform traceroutes to all the representative targets. In addition, I schedule probes from each public traceroute server to a small random set of BGP prefixes, making a few hundred measurements from each server. The public traceroute servers serve as a valuable source of information regarding local routing policies. Note that in the long run, a functioning information plane may actually serve to decrease the load on the public traceroute servers as the information plane, rather than the traceroute servers themselves, can be consulted for information on the Internet topology.

4.4 Path Composition

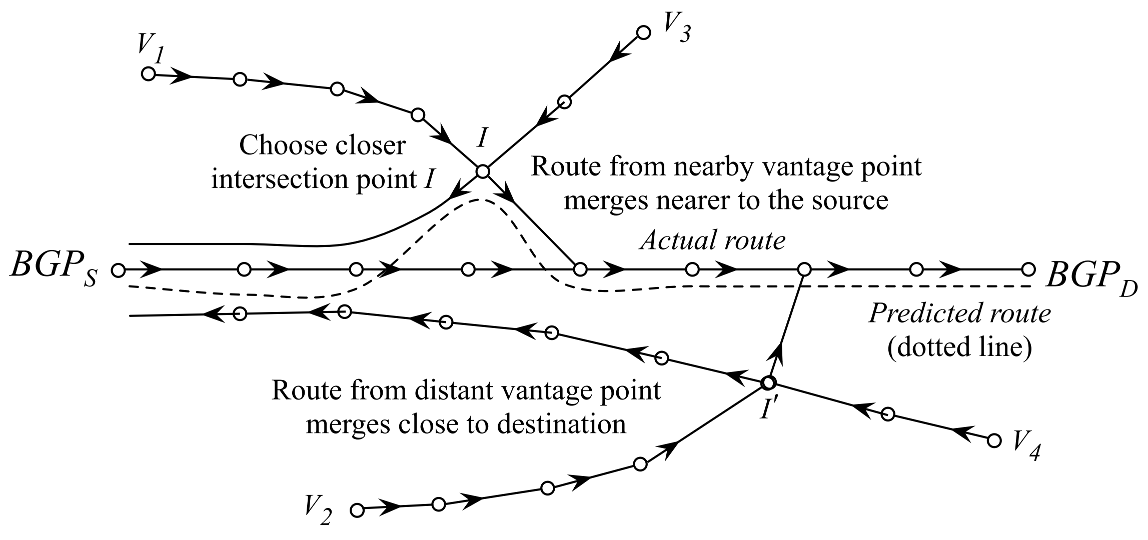
I first consider a setting where the source and destination nodes are passive, but known to the atlas. In other words, the atlas contains observed paths from the vantage points *to* both nodes, but not *from* the nodes.

Figure 4.1(a) depicts how I predict the path from a source S to a destination D using paths from all vantage points to S and D. I seek to determine a path from one of the vantage points to D that has a significant overlap with the actual path from S to D. To this end, I would like a path that originates at S and intersects with one of the paths going into D. In the absence of outwards path from S, I estimate the paths out of S by reversing paths from the vantage points into S. Thus, I traverse paths going into S backwards until I find an intersection I with one of the paths going into D. The predicted path from S to D is obtained by splicing the segment from S to I with the segment from I to D .

The atlas gathered by the vantage points includes traceroutes to only one destination in each routable prefix. Hence, when predicting the path between a pair of end-hosts, the atlas may not include paths to either end-host. In such cases, I approximate the path between S and D with the path predicted between the representative end-hosts in the same prefixes as S and D.

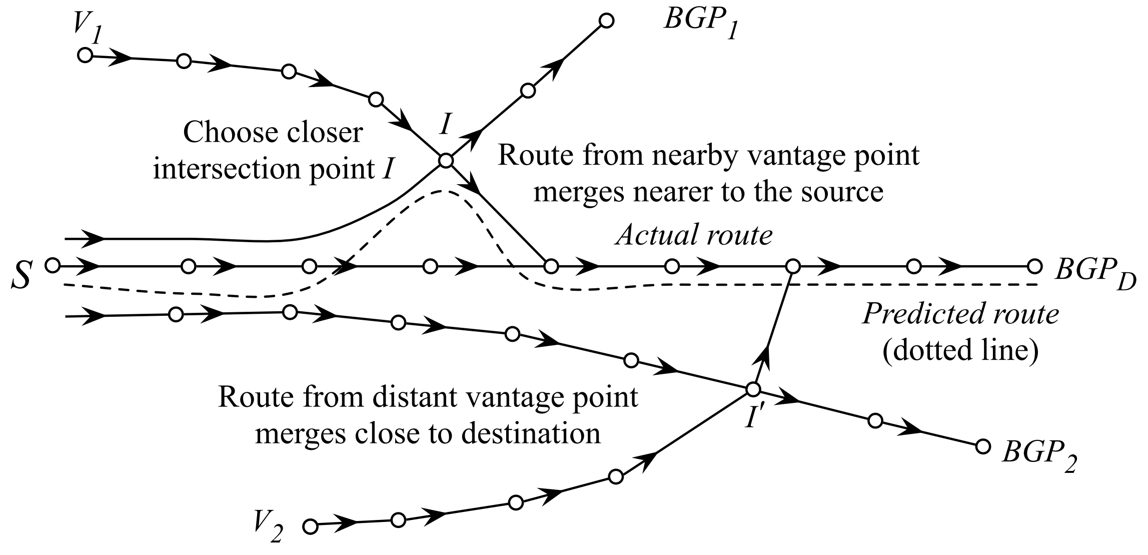
However, predicting outward path segments near the source by reversing paths from the vantage points to the source is not always accurate. A significant fraction of Internet paths are known to be asymmetric [80, 81]. Internet routing is controlled by ISP policy, and ISPs are not constrained to using symmetric policies.

To account for routing asymmetry, I next consider a setting where the source issues a small number of traceroute probes in order to integrate itself into the Internet atlas constructed by the vantage points. A client can be asked to issue probes to a random subset of the vantage points or to representative targets in a small randomly chosen set of BGP atoms. I choose the latter alternative in order to be more representative. As shown in Figure 4.1(b), path segments from these probes are then composed with path segments from vantage points to destinations to obtain the predicted forward path. Of course, since I want to predict path properties from a source to any arbitrary destination, an active client



V_1
V_3
I
Choose closer intersection point I
Route from nearby vantage point merges nearer to the source
Actual route
BGP_S
BGP_D
Predicted route (dotted line)
Route from distant vantage point merges close to destination
I'
V_4
V_2

(a)



V_1
BGP_1
I
Choose closer intersection point I
Route from nearby vantage point merges nearer to the source
Actual route
S
BGP_D
Predicted route (dotted line)
Route from distant vantage point merges close to destination
I'
BGP_2
V_2

(b)

Figure 4.1: The route from S to D is obtained by composing either (a) a route going into BGP_S, a host in the same BGP prefix as a passive client S, or (b) a route contributed by an active client S with a route to BGP_D, a host in the same BGP prefix as D, from a vantage point close to S (V_1).

only helps the forward path—I use passive techniques for the reverse path in the absence of traceroutes from the destination in the atlas.

4.5 Clustering Routers

The path composition method has a few potential problems. The atlas comprising measurements from vantage points to one destination in each prefix and a few measurements from sources does not comprehensively capture Internet routing. As a result, path composition may not always be able to identify intersections between paths. First, the paths obtained using a small number of source probes may not intersect with paths going into the destination. Second, even if a path from the source intersects with a path from a vantage point to the destination at a physical router, the two paths may not appear to intersect. The reason is that traceroutes return information at the granularity of network interfaces, so two paths entering and exiting a router through different network interfaces will appear to not intersect in the resulting graph. It is even harder to determine intersection when two paths pass through the same PoP but through different routers. As a result, the composition method may obtain a grossly inflated path, e.g., a path going back from the source all the way to a vantage point and then following the path going into the destination, similar to the IDMaps [24] method.

While one approach to fix the above problems would be to gather a more comprehensive atlas, I instead argue that routing information at a coarse granularity suffices to capture Internet performance. Striving for efficiency, I attempt to model Internet routing at the granularity that enables the path composition technique to work without information loss. For this, I leverage the property that the path composition technique allows intersections between paths to be determined at various granularities. Declaring two paths to have intersected only if the same network interface address is seen on both is almost certainly too strict for the reasons outlined above. On the other hand, declaring an intersection if both paths traverse a common AS is prone to significant prediction error. Thus, intersections need to be determined at a level coarser-grained than network interfaces but finer-grained than ASes.

I address these problems using a clustering method that groups network interface addresses that are "nearby" from a routing perspective. My objective is to partition all observed router interfaces into clusters and declare an intersection between two paths only if they go through the same cluster. The clusters represent routers that are similar from a routing perspective, e.g., aliased network interface addresses belonging to the same router, routers belonging to the same PoP, and routers that belong to the same AS and are also geographically nearby. To avoid incorrectly predicting transit between ASes, I only cluster routers that belong to the same AS. There is a trade-off between the utility of clusters and their accuracy. Increasing cluster sizes initially improves prediction accuracy by detecting missed intersections but subsequently degrades accuracy by predicting non-existent paths. My clustering algorithm attempts to hit the peak of this curve.

4.5.1 Alias resolution

Every router has multiple router interfaces, which are called aliases of each other. Packets received on any of these interfaces are forwarded using the same routing table. Hence, all interfaces on a router are identical from a routing perspective.

I cluster router interfaces into actual routers by resolving aliases. Interfaces that are potential alias candidates are identified using two different techniques. Employing the Mercator [30] technique, UDP probes are sent to a high-numbered port on every router interface observed in traceroutes. When a router receives a probe destined to one its interfaces, a response is sent back with the source address set to the address of the interface on which the response is sent out. Therefore, interfaces that return responses with the same source address are considered as possible aliases. This also implies that the same router can return responses with different source addresses when probed from different vantage points. Responses other than port-unreachable ICMP error messages, the expected response to my probes, and those with the source addresses in private prefix ranges (192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, and 169.254.0.0/16), because such addresses are valid only locally, are discarded. I then construct a graph with an edge between each router interface that returned a valid response and its associated source address. Every pair of interfaces in the same connected component in this graph are potentially on the same physical router. Sometimes two interfaces return responses with the same source address to UDP probes but are not aliases; this can occur with incorrectly implemented firewalls that return a ICMP PORT UNREACHABLE error message with their own address, instead of spoofing the address of the host that is being probed.

In addition to the Mercator technique, I also identify candidate alias pairs using the fact that interfaces on either end of a long-distance link are usually in the same */30* prefix [32, 102]. More concretely, given two successive IPs x and y along a path, I consider the IP that is in the same /30 prefix as y to be potentially aliased to x.

Finally, I test all the candidate pairs yielded by the above two techniques to determine if they are indeed aliases. Pairs that respond with similar IP-ID values to UDP probes and also respond with similar TTLs to ICMP probes are deemed to be aliases.

In one of my typical measurement runs, of the 4.6M alias candidate pairs yielded by the Mercator technique, 960K pairs were determined to be aliases. The 388K additional alias candidate pairs obtained using the */30* heuristic yielded another 100K alias pairs.

4.5.2 Intra-AS clustering

Apart from different router interfaces on the same router, multiple routers that are geographically nearby are also likely to have similar routing tables. Different routers in the same PoP within an AS will be typically connected to each other with low-latency high-bandwidth links. As a consequence, ASes would assign low IGP weights to intra-PoP links, which would result in multiple routers in the same PoP to compute similar routing tables for themselves. Therefore, in my quest to cluster together routers similar from a routing perspective, I group routers that reside in the same PoP.

I use two different techniques to cluster geographically nearby routers within every AS. First, I determine the DNS names assigned to as many network interfaces as possible. I then use two sources of information—Rocketfuel's `undns` utility [81] and data from the Sarangworld project [71]—to determine the locations of these interfaces based on their DNS names. This step alone does not suffice for clustering geographically co-located interfaces because: 1) several interfaces do not have a DNS name assigned to them, 2) rules for inferring the locations of all DNS names do not exist, and 3) incorrect locations are inferred for interfaces that have been misnamed. For IPs whose locations can be inferred from DNS names, I validate the locations by determining if they are consistent with the measured delays from traceroutes [40]. I probe every interface using ICMP ECHO packets, and deem interfaces that have an RTT smaller than what would be required if packets travel at $\frac{4}{9}^{th}$ the speed of light, a threshold empirically determined in [40], to have incorrectly determined locations.

Second, to cluster interfaces for which a valid location was not determined, I develop an automated algorithm that clusters router interfaces based on the responses received from them when probed from a large number of vantage points. I use the TTL value in the ICMP ECHO-REPLY received in response to my probes. I estimate the number of hops on the reverse path back from a router to the vantage point by guessing the initial TTL value used by the router. As in [52], I estimate the initial TTL corresponding to a received TTL value of TTL_1 to be $TTL_0 = \min\{255, 32 \cdot \lceil TTL_1/32 \rceil\}$, and then estimate the number of hops on the reverse path as $(TTL_0 - TTL_1 + 1)$.

I hypothesize that routers in the same AS that are geographically nearby will take similar reverse paths back to the vantage point from which I probe them, and routers that are not co-located will not display such similarity, e.g., routers in adjacent PoPs at either end of a long-distance link. I verify this hypothesis as follows. Each interface is associated with a *reverse path length vector* that is the vector with as many components as the number of vantage points, and the i^{th} component is the length of the reverse path from the interface back to the i^{th} vantage point. I then determine the L1 distance between the reverse path length vectors of aliases that I determined. Note that the L1 distance between two vectors is the sum of absolute differences between corresponding components. I call the L1 distance divided by the number of components, i.e., the normalized L1 distance, as the cluster distance.

Figure 4.2 shows that less than 2% of alias pairs have a cluster distance greater than 1. This result is expected because irrespective of which interface on a router is probed, the routing of the response is done based on the same routing table. Some error is to be expected because the data for this graph was obtained by probing interfaces over the course

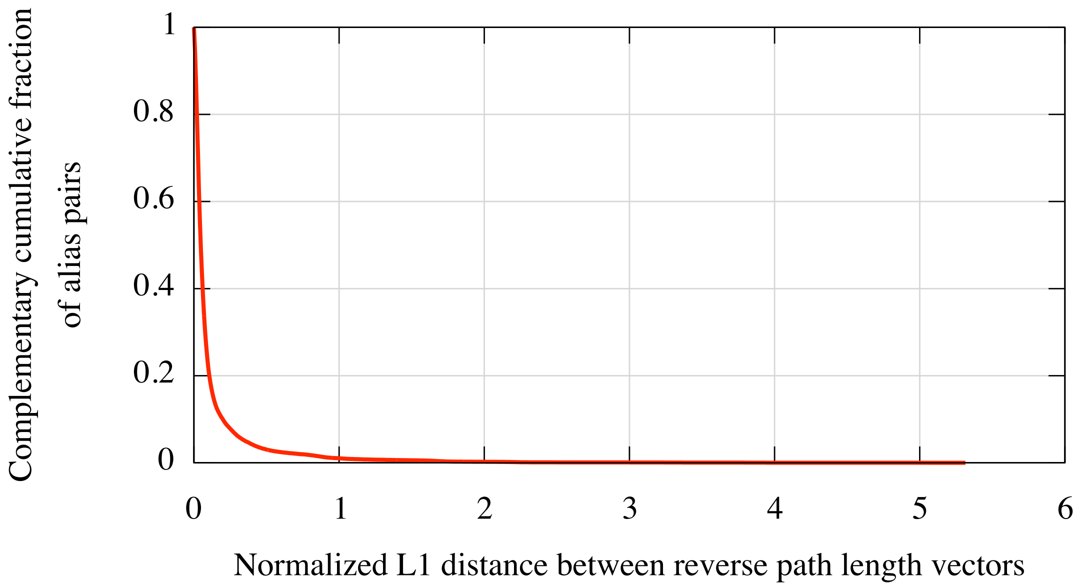


Figure 4.2: Comparison of reverse path length vectors between pairs of interfaces that I determined to be aliases. The reverse path length vector of an interface contains the estimated lengths of the reverse paths back from the interface to all vantage points. The similarity between two vectors is computed as the L1 distance normalized by the number of components.

of a day, rather than instantaneously. The high similarity between reverse paths of aliases implies that route changes over the course of a day are minimal.

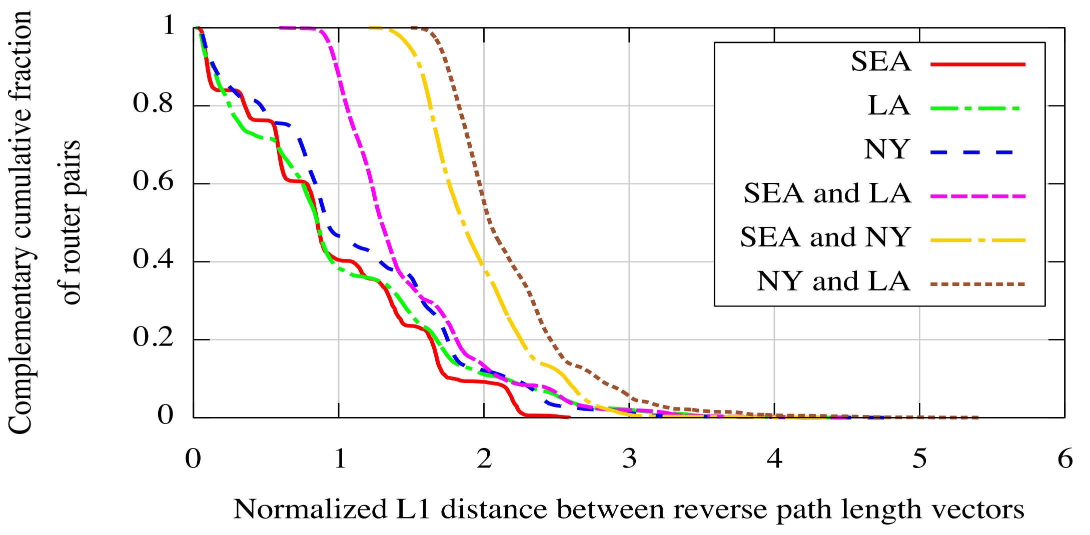
Encouraged by the above result, I then analyzed different routers that belong to the same AS and are known to be in the same city. Using Rocketfuel's `undns` utility [81], I determined the locations of as many router interfaces as possible. I then chose AS 701 (UUNET) for my analysis, and considered the router interfaces in this AS that were reported by `undns` to be in Los Angeles, New York, and Seattle. After discarding interfaces which were not within a few milliseconds from PlanetLab nodes in these locations, I obtained 216, 254 and 64 routers in Los Angeles, New York and Seattle, that belonged to AS 701.

I computed the cluster distance for every pair of routers in the same city, considering one sample router interface per router and then repeated the computation for every pair of routers in different cities. Figure 4.3(a) shows the distribution of cluster distances computed in each of these cases. For almost all pairs of routers in different cities, the cluster distance is greater than 1. On the other hand, more than half the router pairs within each city have a cluster distance less than 1.

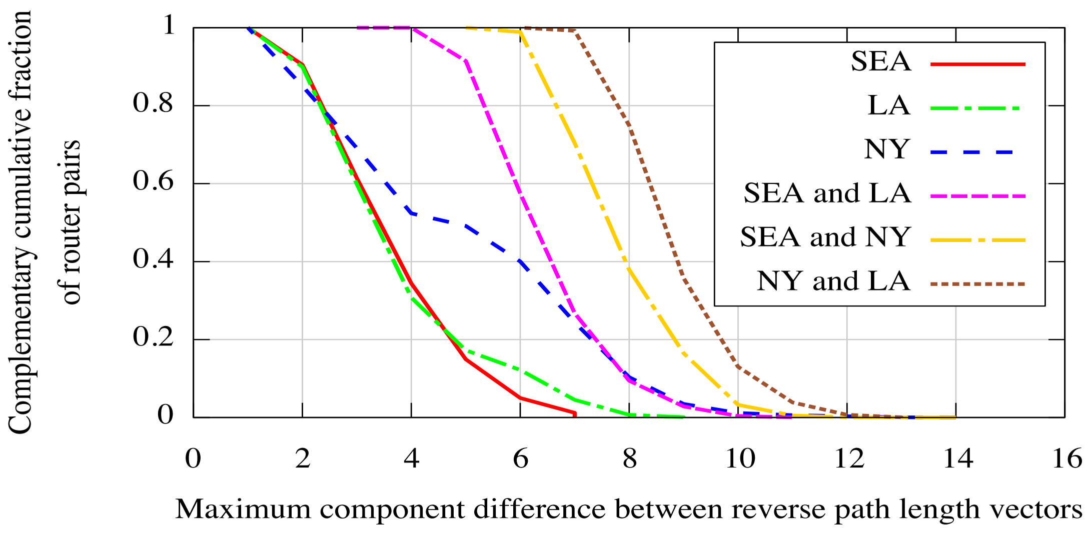
Next, I repeated the above analysis using a different notion of cluster distance defined as the maximum component-wise difference between the corresponding pair of reverse path length vectors. Figure 4.3(b) shows that there is a marked difference between distributions of the cluster distance when routers are in the same city compared to when they are in different cities. Less than 20% of the router pairs in Seattle and LA have a cluster distance greater than 5, whereas for pairs in different cities fewer than 10% are less than that threshold. I believe this clean separation does not hold for routers in NY because of the presence of multiple POPs in that city.

Based on similar results obtained with analysis of reverse path length vectors of routers in other cities, I use thresholds of 1 and 5 for the average and maximum component-wise differences in my clustering algorithm. Though these thresholds were determined using the sample data presented for UUNET and for a couple other ASes, I assume these thresholds are representative of all ASes. Routers within an AS are grouped into clusters such that any pair of routers in the same cluster have cluster distances of at most 1 and 5 respectively using the two definitions. Although these thresholds split some router pairs in the same city into different clusters, even in the cases considered in Figure 4.3, this is in keeping with my principle of erring on the side of accuracy.

Of the 3M router interfaces observed in all of my traceroutes, aliases were determined for 286K of them. These 286K interfaces were grouped into 67K routers. Further, 1.9M interfaces yielded DNS names, of which locations could be determined and validated for 700K of them, given the DNS name to location mapping rules that I used. By enforcing the constraints imposed by aliases and DNS based locations and then applying TTL-based clustering, 2.3M interfaces were grouped into 134K clusters; the remaining did not yield a DNS name and did not respond to any probes. 475K interfaces are in 215 clusters of size greater than 1000, 1.4M interfaces are in 3.7K clusters of size greater than 100, 2M



(a)



(b)

Figure 4.3: Comparison of reverse path length vectors between pairs of interfaces in the same AS but either in the same or in different cities. The similarity between two vectors is computed as (a) the L1 distance normalized by the number of components and (b) the maximum difference between corresponding components.

interfaces are in 26K clusters of size greater than 10, and 56K interfaces are in singleton clusters.

4.6 Path Selection

While clustering improves the fidelity of the path composition technique, many pairs of path segments can be composed to predict the end-to-end path. Which path should one pick? Ideally, you would want to choose the path thats models how Internet routing works in practice. To appreciate this claim, consider a naive scheme that composes a pair of path segments whose combined latency is the least. This scheme predicts paths whose latencies are similar to those predicted by coordinate-based systems; the predicted end-to-end path will be shorter than any detour path through the vantage point, resulting in an incorrect model that does not permit detours.

To model Internet routing, I use information obtained from traceroute probes in combination with basic BGP information. I use BGP tables to determine the origin AS for each IP address encountered on a path segment. Each path segment is then assigned a path length in AS hops, a path length in IP hops, and a latency estimate. These metrics are known to largely determine paths used in the Internet. For example, inter-domain routing prefers shorter AS paths over longer ones in the absence of conflicting locally preferred policies, and intra-domain routing uses latency of paths within the AS to determine the appropriate exit point.

4.6.1 Modeling Default Routing Policy

The basic problem at hand is to figure out a scheme to choose among the several intersections between path segments. I considered a handful of schemes that were in keeping with my hypothesis of route similarity and with the normal understanding of how routing works in the Internet. Here, I discuss the two schemes that worked best.

1. **Min predicted AS path length, min RTT to source**:
   Choose the intersection that minimizes the AS path length along the predicted path. Among those, choose the one that has the minimum RTT from the source. Minimizing

overall AS path length approximates BGP's default objective function; by default, BGP chooses the shortest AS path. Minimizing RTT from the source, an indicator of the intersection being geographically close to the source, captures the principle of route similarity.

2. **Min predicted AS path length, early exit**:
   Choose the intersection that minimizes the AS path length along the predicted path. Among those, choose the one that results in the minimum RTT from the source to the exit point from the source's AS. This policy directly emulates early exit routing, a common intra-AS routing policy.

4.6.2 Export Policies

I also attempt to model routing policy more directly by inferring export policies of ASes from the atlas of observed paths. For instance, if I observe a path that traverses the ASes Cogent, AT&T, and Sprint in succession, I know that AT&T exports paths from Sprint to Cogent.

I incorporate these inferred export policies into the path composition technique using the *3-tuple check*. I explicitly store the list of all 3-tuples corresponding to three consecutive ASes observed in traceroutes as well as BGP feeds (discounting prepending). Ideally, I would consider a candidate intersection as yielding a valid path only if the 3-tuple *(AS preceding the intersection, AS of the intersection cluster, AS succeeding the intersection)* is present in the list of observed 3-tuples; all other sequences of ASes need not be verified since the path segments used for composition were both observed. This implementation of the 3-tuple check ensures that a candidate path is considered only if the export policy of the intersection AS permits it.

However, since visibility into ASes at the edge is limited, I might fail to observe all of the export policies for the edge ASes. I thus perform this check only for 3-segments in which the degree of the intersection AS in the Internet's AS-level graph, obtained from inter-AS edges observed in traceroutes and BGP feeds, is greater than a threshold. I find a threshold value of 5 to be a sweet spot in accounting for the lack of observations while ensuring enforcement of export policies.

I also assume commutativity among triples, so that if I observe (AS1, AS2, AS3), I include (AS3, AS2, AS1) as well in the list of 3-tuples. This is because AS paths give the path only in one direction and I have two orders of magnitude fewer vantage points than there are ASes in the Internet.

4.7 Optimization of Atlas Size

The size of the raw atlas of traceroutes gathered from vantage points to destinations in all prefixes is typically of the order of 1-2GB after compression. When loaded into memory, the size of the atlas inflates further to around 4GB because of the overhead of the data structures into which the atlas is read.

To optimize the storage and representation of the atlas, I present a simple algorithm, CLUSTER-TREE, which leverages the property that the path composition technique only needs path information at the cluster granularity. CLUSTER-TREE stores cluster-level trees to each destination. It leverages the insight from Doubletree [21] that paths from vantage points to a common destination share many links due to the destination-based nature of Internet routing. Storing cluster-level trees instead of complete routes to each destination does not affect the accuracy of the path composition algorithm. However, the size of the cluster-level tree is just 300MB compared to 3GB for the raw atlas—an order of magnitude reduction in size.

4.8 Evaluation

In this section, I describe the results of the experiments I performed to evaluate the algorithms presented in this chapter. First, I outline the measurements I performed to gather an Internet atlas and to cluster the router interfaces observed in this atlas. Thereafter, I present results validating the paths predicted by the path composition technique. The accuracy of path prediction is crucial for accurate estimation of path properties since these are computed by composing properties of links and path segments along the predicted route.

4.8.1 Measurements

I use nodes in PlanetLab [65], and several hundred public traceroute and Looking Glass servers to perform my measurements. My dataset consists of two distinct sets of measurements: one for building the atlas that my model uses for prediction and the other for evaluating the model. There are no end-to-end paths that are common between the two sets.

The measured atlas comprises measurements made from 158 PlanetLab nodes to a wide range of destinations. I obtained the list of all globally routable prefixes from the BGP snapshot gathered by RouteViews [53]. In each of these prefixes, I determined a *.1* address that responded to either ICMP or UDP probes. I compiled a list of 87334 destinations, with no two of them being in the same prefix. On 3 February 2006, I performed traceroutes from 158 PlanetLab nodes to all these destinations, to the public traceroute and Looking Glass servers, and to the PlanetLab nodes themselves.

To evaluate my model, I use a validation set comprising paths measured from the public traceroute and Looking Glass servers. On 3 February 2006, I performed traceroutes from each of these traceroute servers to 100 other randomly chosen traceroute servers and to destinations in 200 random prefixes chosen from the prefix list. In these traces as well as in those gathered from PlanetLab, I ignore all paths that either did not reach the destination or had a router-level loop. I note once again that this validation set shares no paths with the routes measured from PlanetLab nodes that I use to build the atlas; however, I do use these traceroutes to understand in detail why my path predictions are accurate/inaccurate. Using paths measured from PlanetLab nodes as my atlas, I evaluate path prediction on the paths measured from traceroute and Looking Glass servers.

4.8.2 Path Composition: Feasibility Analysis

The path composition technique yields several candidate paths between a source and a destination. I first evaluate how closely any of these candidates matches the observed path. Though the path composition technique returns paths at the granularity of clusters, comparison of cluster-level paths is non-trivial. This is because my clustering is not perfect;

two router interfaces in different clusters could in fact be in the same PoP of an AS but I could have failed to identify them as being so. Therefore, I restrict my evaluation of path prediction to AS-level paths, leaving the comparison of cluster-level paths for future work.

To compare the actual and predicted AS paths, I define an AS path similarity metric that is similar to the RSIM metric used in [34]. I define the similarity metric between two AS paths to be the ratio of the size of the intersection, to the size of the union, of the sets of ASes in each of the paths. The maximum value this metric can take is 1, which is when both paths pass through the same set of ASes.

For each source-destination pair, I considered all intersections obtained using the basic technique of intersecting paths from/to the source with paths to the destination in the atlas. I refer to the policy that chooses (using the validation data set) the intersection such that the predicted path passing through it maximizes the AS similarity metric as the *optimal policy*. Figure 4.4 plots the distribution of the similarity metric for this optimal policy. The policy *Optimal (k)* evaluates the scenario with *k* paths from each validation source present in the atlas.

Figure 4.4 shows that, in the absence of any paths from the source, around 40% of paths are such that none of the candidate paths yielded by path composition has the same AS path as the actual path. This implies that irrespective of the policy I use to choose among intersections between paths going into the source and into the destination, in a large fraction of cases, I cannot predict a path that has the same AS path as the actual path.

A potential reason for this is that paths going into the source may not be representative of paths coming out from it. To confirm this hypothesis, I evaluate the feasibility of finding the actual path in the case of active clients wherein the atlas contains a few paths originating from each end-host. I chose a few paths from each Looking Glass server at random and moved these paths from my validation set into the atlas. For each of the paths remaining in the validation set, I considered intersections of the chosen few paths from the source with all the paths from PlanetLab nodes to the destination. In cases where there was no intersection, I fell back to the passive client model and used paths going into the source.

Figure 4.4 shows that using a few paths originating from the source significantly improves the number of cases in which at least one of the candidate paths yielded by path composition

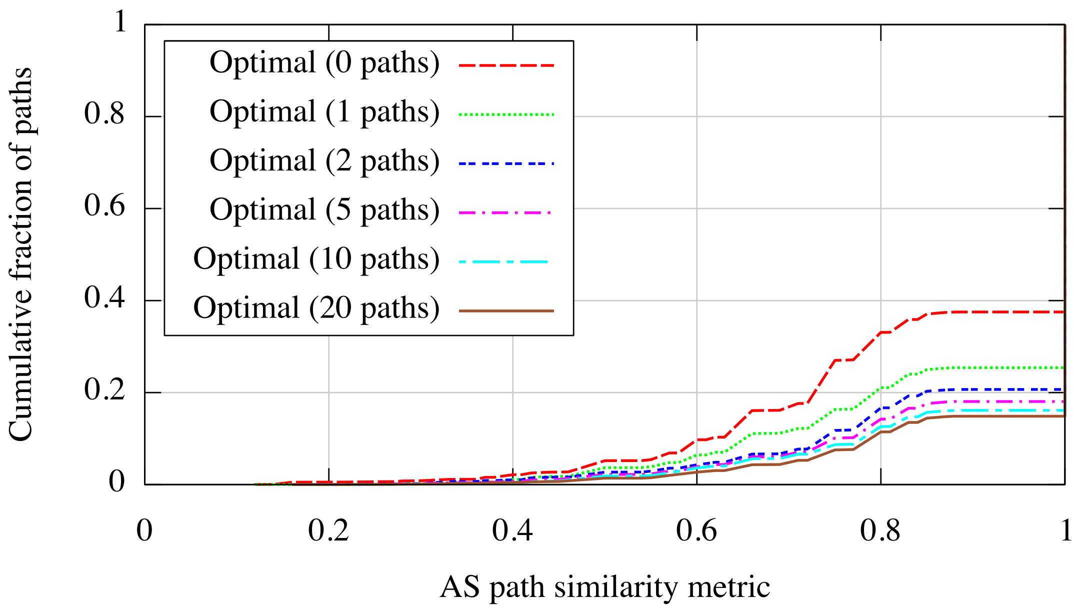


Figure 4.4: Evaluation of the feasibility to predict AS paths using the path composition technique. *Optimal (k paths)* is an oracle that determines the path most similar to the actual path between a source and a destination using the path composition technique, given the presence of k paths from the source in the atlas. The similarity between two AS paths is computed as the ratio of the size of the intersection, to the size of the union, of the sets of ASes in each of the paths.

has an AS path identical to the actual path. Availability of 10 paths from the source to random destinations increases the fraction of paths for which the optimal policy could get the AS path exactly right from just over 60% to over 80%. This shows that adding a few paths from the source to the atlas significantly improves the likelihood of the actual path being one of the candidate paths obtained with path composition. Figure 4.4 also shows that the marginal increase in the ability to perform AS path prediction with the use of more than 10 paths is small. So, for the rest of the evaluation, I assume that I have 10 paths from each source.

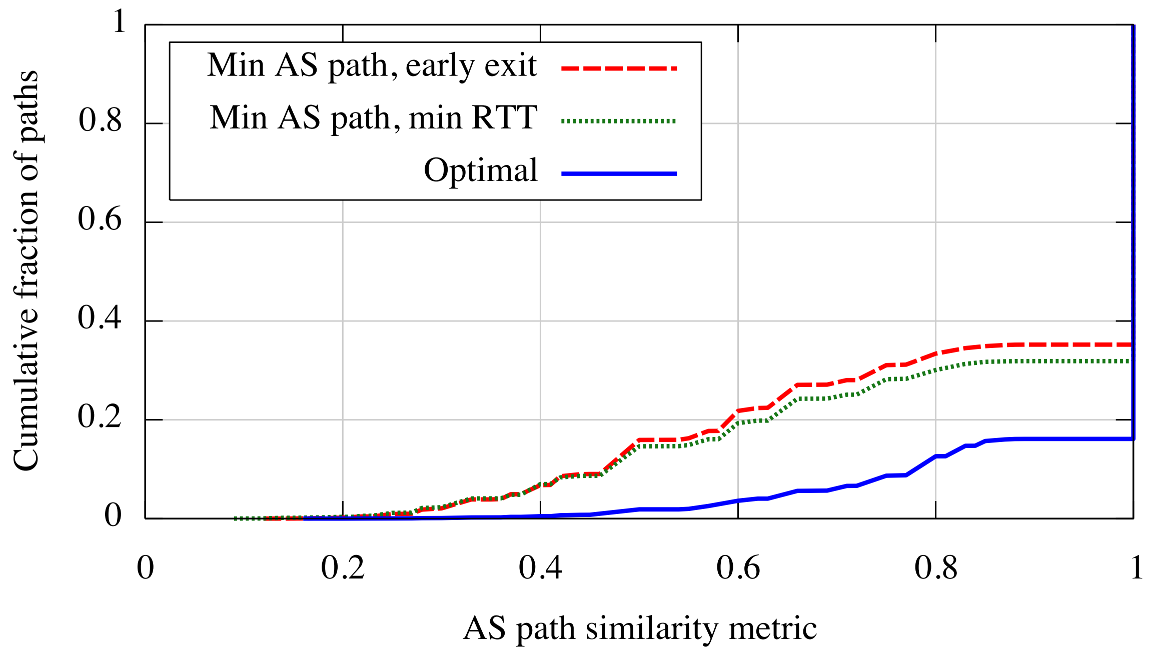


Figure 4.5: Evaluation of the *Min AS path, min RTT* and *Min AS path, early exit* policies for selecting among the candidate paths yielded by path composition. The AS path prediction accuracy of both policies is compared against the optimal.

4.8.3 Path Prediction

The previous results just demonstrate that it is *possible* to predict the actual path when I have access to a few paths from the source. I next evaluate how well my incorporation of routing policy into the path composition technique helps choose the actual path among candidate options, if it exists at all.

How does default routing policy perform?

First, I consider the two schemes that I use to model default routing policy—*Min AS path, min RTT* and *Min AS path, early exit*. The accuracy with which these schemes help in finding the correct AS path, for the case when I have 10 traceroutes from the source, is shown in Figure 4.5. While neither is perfect, approximating BGP's shortest AS path policy and early exit routing helps identify the correct AS path in more than 65% of cases.

Minimizing the AS path length is more likely to choose a policy compliant path, and by choosing an intersection close to the source I minimize the number of routing policy decisions that I predict, and hence, am likely to make fewer incorrect routing choices. Since the path segment from the point of intersection to the destination is an observed one, there cannot be any incorrect decisions made along that segment.

Do predicted paths violate AS relationships?

Since my model does not involve use of inferred AS relationships, the paths I predict could violate policy routing. To see if this is a problem, I applied Gao's AS relationship inference algorithm [27] on AS paths observed in all my traceroute measurements as well as on AS paths obtained from the RouteViews [53] BGP dump. I then computed how many of the AS paths in my validation set, and how many of the predicted AS paths (using the *min AS path, min RTT* policy), violate valley-free routing. Of the 42028 paths in the validation set, valley-free routing was violated in 119 of the observed AS paths and in 212 of the predicted AS paths. This demonstrates that my incorporation of routing policy into the composition technique does usually manage to find policy-compliant paths. As noted earlier, the choice of an intersection close to the source minimizes the number of routing policy decisions my model must predict. This result also implies that constraining my model by explicitly inferring AS relationships and enforcing valley-free routing using these relationships on the candidate paths will not decrease its predictive ability.

I leave the evaluation of the benefit of incorporating AS export policies into the path composition technique for Chapter 5 since I also use them in the graph-based path prediction model that I develop in that chapter.

4.9 Summary

The ability to predict paths between arbitrary end-hosts on the Internet is one of the key pieces towards building an Internet-wide information plane using a structural approach. In this chapter, I proposed a model for path prediction using an atlas of Internet routes gathered from several hundred geographically distributed vantage points. The key technique I use to predict routes is to compose segments of observed paths. The path composition technique is based on the principle of route similarity—routes from nearby sources to the same destination are similar.

I recognize and demonstrate that this path composition technique can work without requiring the atlas to comprehensively capture all routing information in the Internet. First, I minimize the measurement load on the vantage points by restricting traceroutes to one representative end-host in every prefix or every BGP atom. Second, I cluster router interfaces that are similar from a routing and performance perspective. Figure 4.4 shows that these techniques to account for the lack of comprehensiveness of the atlas enables the path composition technique to yield an AS path identical to the actual path in over 80% of cases.

However, composing path segments can yield several candidate paths between a source and a destination. To choose among these candidates, I both model default routing policy and enforce routing export policies of ASes inferred from observed paths. This enables the path composition technique to predict the AS path between end-hosts exactly right in more than 60% of cases, with the addition of as few as 10 paths from each end-host into the atlas.

Chapter 5
GRAPH-BASED PATH PREDICTION

Prediction of paths by composition of path segments requires as input an Internet atlas of observed paths. The size of such an atlas is proportional to the number of vantage points times the number of destinations probed times the average path length. My atlas comprising paths from all PlanetLab nodes to all BGP prefixes is over 1GB in size even after compression. As more vantage points contribute measurements, the accuracy of paths predicted by composing path segments should increase, but at the cost of blowing up the size of the atlas.

In this chapter, I develop a path prediction model that makes predictions by composing observed links, in contrast to techniques presented in the previous chapter that compose measured path segments. Stitching together path predictions at the link granularity has the potential to result in maps with size proportional to the number of observed links, as opposed to the number of measured paths, enabling a more compact representation. The compact size lowers storage costs and distribution bandwidth overheads, making it feasible to distribute daily maps to low bandwidth clients.

My approach towards developing a model for predictions at link granularity is to start off with the Internet graph seen in union across all path measurements in my atlas of paths. Since routing in the Internet is destination-based, it should be possible to encode data as decisions at each node in the graph. I start with a very simple model of router behavior based on commonly properties of Internet routing. I then augment this behavior by extracting routing policy from observed paths and storing this policy in a compactly to improve prediction accuracy without sacrificing efficiency.

However, such a fine-grained approach has its downsides. In order to predict an end-to-end path accurately, I need to accurately model the routing behavior at every router along the path, a particularly challenging task for backbone routers with a diversity of network connections. The challenge, therefore, is to preserve prediction accuracy under fine-grained link composition.

5.1 The Problem: Modeling Internet Routing

The problem of predicting Internet routes would be trivial if routers used shortest path routing. Even if the weights used in a shortest path calculation are unknown, one could infer them by making a sufficient number of observations of actual routing behavior, as has been done with intra-domain OSPF weights in [48]. However, Internet route selection includes a number of factors, such as monetary costs, coarse-grained path metrics, and local performance considerations. More importantly, these factors vary from ISP to ISP, and are often expressed in a policy language that allows for many special cases. The issue is further complicated due to the interaction between intro-domain and inter-domain routing policy (as shown in Chapter 2).

Thus, the challenge is to develop a model of Internet routing that captures most of its complex behavior while working on a link-level representation of the Internet atlas. Fortunately, previous work on reverse-engineering the routing decision process aids me in this effort, as does the body of knowledge the research community has accrued on how Internet routing works in practice.

My goal is to develop a procedure that incorporates the following commonly accepted principles regarding how routing works on the Internet.

1. *Policy preference*: ASes use *local preferences* to select routes. Typically, an AS prefers routes through its customers over those through its peers, and either of those over routes through its providers. Further, ASes do not export all of their paths to their neighbors; for instance, ASes do not export paths through their peers to other peers/providers. Commonly used export policies and AS preferences are believed to result in *valley-free* Internet routes [27], in which any path that traverses a provider-to-customer edge or a peer-to-peer edge does not later traverse a customer-to-provider or peer-to-peer edge.

2. *Shortest AS path*: After considering local preferences, if a router has multiple candi-

date paths that it prefers equally, the default behavior is to select the route containing the fewest ASes. Typically, several paths may have the same AS path length.

3. *Early-exit vs. Late-exit*: Among these, routes are chosen so as to meet intradomain objectives, e.g. by choosing the nearest exit point into the next AS on a remaining path (referred to as early-exit or hot potato routing). In certain cases, especially when the same economic entity owns adjacent ASes, ASes might instead collaborate to reduce their combined costs (typically referred to as late-exit) or to accomplish other goals, such as to reroute traffic to reduce congestion.

I develop an algorithm using the classic dynamic programming technique that underlies various forms of shortest path computation. The algorithm incorporates the above criteria to compute an on-demand route, based on a graph representation of the atlas. My first attempt, GRAPH, reduces the representation size by over two orders of magnitude, but has poor prediction accuracy compared to path composition techniques. I then present a series of refinements designed to address GRAPH's shortcomings and arrive at a technique that stores only a little more information than GRAPH, yet yields prediction accuracy comparable to the path composition approach. Table 5.1 summarizes the various techniques I employ in predicting paths using an atlas of links.

5.2 GRAPH: *A first cut*

Setting: As with the path composition technique, a distributed set of vantage points issues traceroute probes to various prefixes. Instead of storing the set of observed routes, GRAPH stores the IP-level graph corresponding to the Internet's observed routing topology. The nodes of the graph are IP addresses, and its undirected edges are the observed edges in the traceroutes.

In addition, I use data from BGP feeds, such as RouteViews [53] to store a mapping from IP prefixes to their origin AS. I also use the data put together in the previous chapter

Table 5.1: Summary of techniques employed to develop datasets and algorithms used for predicting paths between end-hosts using a link-based atlas.

Technique	**Description**	**Goal**	**Section**
Traceroutes from vantage points	Paths to all prefixes/atoms are measured from a large number of geographically distributed vantage points	Gather graph of inter-IP links	Section 4.3
Dijkstra-style algorithm with two-tuple cost	Every node in the graph is associated with a two-tuple cost (AS path length, intra-AS latency), and path to the destination is found that minimizes cost	Enforce shortest AS path and early-exit	Section 5.2
Two-plane atlas	Links discovered from paths from end-hosts and from paths to all prefixes/atoms are separated out into two graphs	Account for routing asymmetry	Section 5.4
AS three-tuples	Gather all sequences of three successive ASes seen in observed paths	Capture path export policies	Section 5.5
AS preferences	Compare observed paths with alternate policy-compliant paths that were not chosen	Detect local preferences	Section 5.6
Provider mapping	Store provider ASes for prefixes/ASes that do not have all their upstream ASes as providers	Account for traffic engineering	Section 5.7

that identifies which sets of IPs can be clustered together.

Algorithm: I present the algorithm in multiple steps. I first provide a dynamic programming technique (similar to Dijkstra's shortest path algorithm) that captures the preference for short AS paths, with early-exit deployed between every pair of ASes. I then briefly describe how I tweak the algorithm to model late-exit when necessary. I conclude the al-

```
Graph($s, d$):
$N' \leftarrow \{d\}$
for each $v \in G$
  if $v$ is a neighbor of $d$, then $\mathbf{D}(v) = c(v, d)$;
  else $\mathbf{D}(v) = [\infty, \infty]$ ;
Do
  Pick $w \notin N'$ such that $\mathbf{D}(w, d)$ is a minimum
  $N' \leftarrow N' \cup \{w\}$;
  for each neighbor $v$ of $w$
    if $\mathbf{D}(v, d) > \mathbf{D}(w, d) \oplus c(v, w)$, then
      $\mathbf{D}(v, d) = \mathbf{D}(w, d) \oplus c(v, w)$;
      $P(v, d) = v.P(w, d)$;
until $N = N'$
```

Figure 5.1: The Graph algorithm to predict a route from s to d in a graph G.

gorithm description by outlining the modifications required to incorporate common export policies and local preferences for selecting routes.

Figure 5.1 shows the pseudocode for Graph, an algorithm that predicts the route between a source s and a destination d. It chooses the shortest AS path between s and d, while performing early-exit at every AS. The algorithm is similar to Dijkstra's shortest path algorithm. Unlike conventional Dijkstra however, the route computation 1) backtracks from the destination to all sources and 2) uses a two-tuple cost metric.

The cost of a route from each node to the destination is a strictly ordered two-tuple [number of AS hops to the destination, cost to exit the current AS], with the first component considered as the more significant value. For two adjacent nodes v and w connected by a link of latency $l(v, w)$, $c(v, w)$ is defined as $[0, l(v, w)]$ if v and w are in the same AS, and as $[1, 0]$ otherwise. The $\oplus$ operator in the algorithm resets the second component to 0 upon crossing an AS boundary as follows. If v and w belong to the same AS, $\mathbf{D}(w, d) \oplus c(v, w)$ is defined as $\mathbf{D}(w, d) + [0, l(v, w)]$. If v and w belong to adjacent ASes, $\mathbf{D}(w, d) \oplus c(v, w)$

is defined as $[\mathbf{D}(w,d)[1]+1,0]$. It is straightforward to verify that this definition of cost preserves the invariant that if a node $u \in N'$, then $P(u,d)$ is a shortest path from u to d.

I then incorporate constraints corresponding to common export policies. I infer AS relationships, such as which are peers and which have paid customer/provider transit, using a combination of CAIDA's inferences [19] and Gao's technique [27]. I model the default export policy in which an AS advertises any paths through customer ASes to all its neighbors, and it exports its paths through peers and providers to only its customers. It is well-known that this export policy leads to valley-free routes, and I modify the algorithm to compute only valley-free routes to a given destination. To compute valley-free routes, instead of having a single node for each IP address i, I instead introduce two nodes in the graph: an *up* node up_i and a *down* node $down_i$, and Graph computes the path from up_s to $down_d$.

The idea is that the construction of edges will force every path to transition from *up* nodes to *down* nodes at most once, thereby guaranteeing the path is valley-free. Let i and j be two IP addresses observed as adjacent. If i and j belong to the same AS, there is an undirected edge between up_i and up_j and one between $down_i$ and $down_j$. If i's AS is a provider of j's AS, there is a directed edge from up_j to up_i and another directed edge from $down_i$ to $down_j$. Essentially, this configuration captures that a customer will not provide transit between two providers. If i and j belong to peer ASes, there is a directed edge from up_i to $down_j$ and from up_j to $down_i$. These edges capture that i's AS will use paths through j only for itself and its customers (and similarly for j's AS and paths through i). For each IP address i, there is a directed edge from up_i to $down_i$. All routes in the graph are therefore valley-free by construction.

I conclude the algorithm development by modifying the technique to take into account local preferences in adopting AS paths. In particular, I assume that an AS prefers paths through its customers over those through its peers, which are in turn preferable to paths through provider ASes. To simulate this, instead of calculating paths to the destination from all ASes and all routers in a batch, I stage the computation. I first limit the graph to contain only the set of *down* nodes, along with the edges connecting them, and compute the optimal paths from these nodes to the destination. This reaches precisely the routers in those ASes that get paid for providing transit to the destination. Once all such nodes

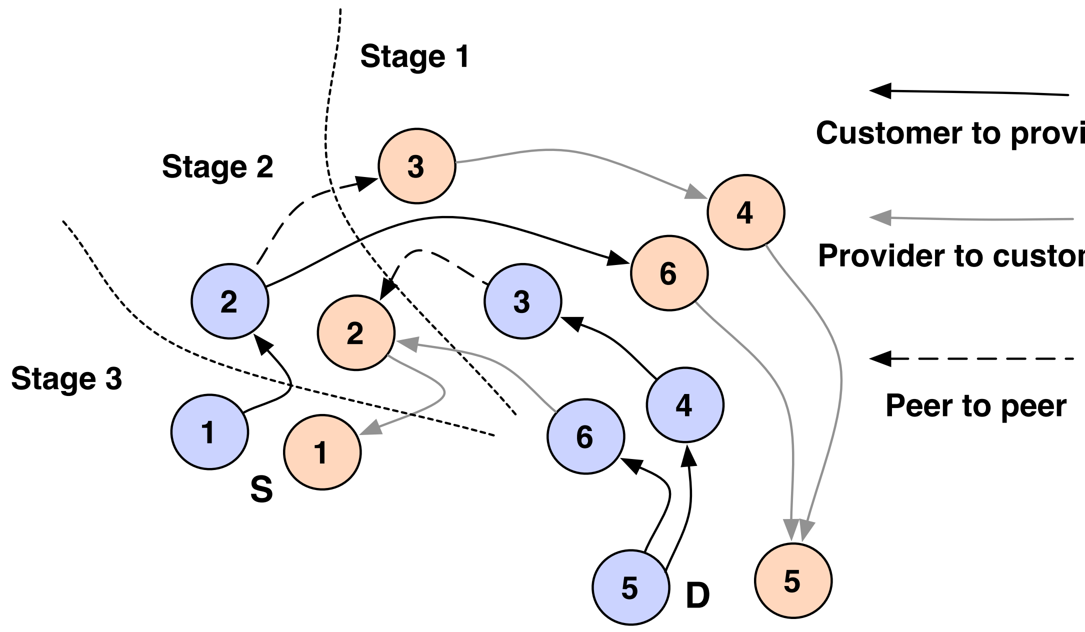


Figure 5.2: Predicting the path from S to D in keeping with customer<peer<provider preferences. Blue nodes are down nodes, and pink nodes are up nodes. Bold lines go from customers to their providers, dashed lines connect peers, and faded lines go from providers to their customers. Graph traverses all the customer-to-provider edges in the first phase to finalize routes from 3, 4, and 6 to D. Only peering links are traversed in the second phase making 2 choose a path through 3 over a shorter one via 6. Finally, edges from providers to their customers are traversed.

have been visited and their best paths discovered, I then allow the algorithm to reach any additional nodes that can be reached only using peering; by construction, only one peering is traversed. Finally, I allow the algorithm to use any link (e.g., provider links) to reach all remaining addresses. This procedure enforces the preference of any AS to attempt to first route through one of its customers, then through one of its peers, and finally through one of its providers. Figure 5.2 illustrates these three phases.

Evaluation: As I show in detail in Section 5.8, Graph—despite taking into account many aspects of default routing behavior—correctly predicts only 30% of the AS paths for my measured dataset, compared with 70% using the path composition technique on the same data. 30% accuracy in predicting AS paths is consistent with results in [67].

On the other hand, the storage overhead of Graph is directly proportional to the number of observed Internet links. In practice, as I demonstrate in the evaluation section, this is two orders of magnitude more compact than the path composition approach. Thus, the challenge is improving Graph's accuracy while keeping its storage advantages.

5.3 Sources of Prediction Error

Graph's inaccuracies arise partly from its failure to model certain aspects of Internet routing behavior and partly from errors in inferred AS relationships. In particular, Graph's deficiencies are due to the following reasons:

1. *Asymmetry*: A significant fraction of Internet routes are asymmetric [63]. While Graph reflects some asymmetry, e.g., due to early exit routing, it does not capture the full range of asymmetric policy behavior.

2. *Inaccurate export policy*: If Graph fails to identify a peer-to-peer relationship between a pair of ASes, then it would allow an overly lenient export policy, predicting non-existent routes that will be filtered in practice.

3. *Incorrect local preferences*: An AS's customer may be a provider for specific paths, and incorrect local preferences could result in an AS selecting a less preferable route, e.g., via a customer.

4. *Traffic engineering*: ASes may engineer routes so as to ensure that their customer traffic use better routes, e.g., to avoid congestion.

The next four sections describe how I address each of these challenges to improve upon Graph.

5.4 Addressing asymmetry

In the previous chapter, due to the asymmetric nature of Internet routing, I observed that path prediction accuracy significantly improves with the addition of paths originating *from* the source to the atlas. Similarly, in this case, I add links observed on paths from end-hosts to the graph atlas. To reduce the likelihood of predicting non-existent routes, I split the graph into two subgraphs: TO_DST consists of all directed links observed on traceroutes from vantage points to all prefixes, and FROM_SRC consists of all directed links on traceroutes from participating sources. For each IP address, I introduce a directed edge from its corresponding node in FROM_SRC to its corresponding node in TO_DST. I then predict the route using the Dijkstra-style algorithm used in GRAPH that backtracks from the destination node in TO_DST to the source node in FROM_SRC. If I fail to find such a route, a likely scenario if the atlas lacks sufficient paths from the source prefix, then I attempt to find a path from the destination node in TO_DST to the source node in TO_DST. For both the source and the destination, their absence from either plane is handled by considering an IP address in the same prefix instead.

The technique described above transitions between the two planes only at a single IP, but I also allow transitions between two IPs on the same router or in the same PoP since in a given AS, routers in a PoP typically have similar routing. For this, I use the clustering of router interfaces described in the previous chapters and introduce directed edges between pairs of IPs belonging to the same cluster. For each pair of nodes x and y such that x belongs to FROM_SRC and y belongs to TO_DST and x and y are in the same cluster, there is a directed edge from x to y. It is straightforward to verify that, by construction, routes in this graph transition from FROM_SRC to TO_DST at most once. In the current implementation, I limit any route to at most one unobserved transition within any cluster. This can be further extended by representing each cluster as a single graph vertex. Figure 5.3 illustrates the two-plane approach.

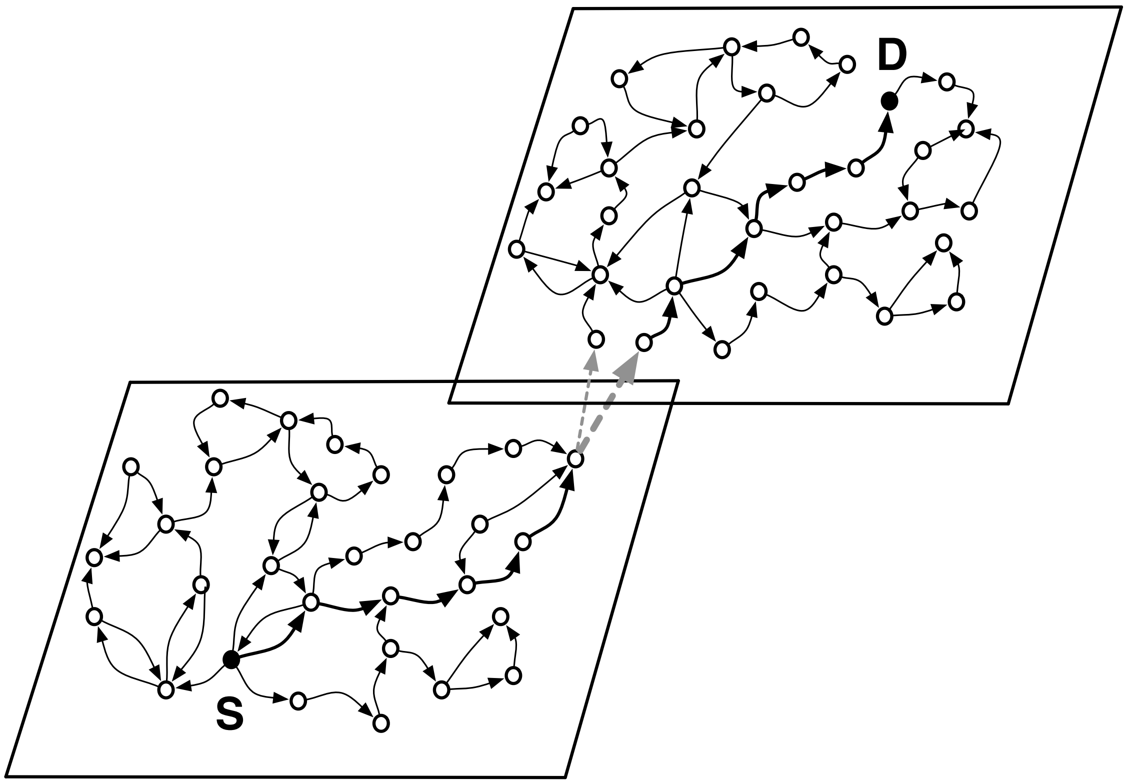


Figure 5.3: Atlas of links is divided into two planes to account for routing asymmetry. One plane comprises links measured on paths from vantage points to all prefixes and the other contains links seen on paths contributed by end-hosts.

5.5 Export Policies

As mentioned above, Graph suffers from the problem of predicting some non-existent routes, especially routes that would be filtered given accurate AS relationships. Therefore, instead of explicitly distilling the AS relationships from the observed routes, I instead use the export policies inferred in Section 4.6.2. The three-tuple check that incorporates the inferred export policies is enforced as follows. During path prediction, I backtrack from AS2 to its neighbor AS1 only if AS2's already determined successor towards the destination, say AS3, is such that the three-tuple (AS1, AS2, AS3) was seen on one of the observed paths.

5.6 AS preferences

Recall that I infer AS relationships and incorporate the customer<peer<provider preference order in the route prediction algorithm. Unfortunately, AS relation inference by itself is difficult and error-prone. For example, AS relationship inference based on Gao's algorithm [27] predicts that half of the edges observed between the top hundred ASes ranked by degree correspond to sibling relationships, which seems rather implausible. The 3-tuple check by itself is not sufficient; although it ensures that predicted routes consist only of observed tuples, it does not take AS preferences into account when multiple options are available.

I use a relationship-agnostic method to infer AS preferences based only on observed routes. I infer these preferences using the entire set of observed paths, but I include only the results of the inferences within the compressed link-level representation of the atlas. The technique works as follows. For each observed AS route r, let $r_1, \ldots, r_m$ be the set of alternative routes available from the source (I discuss how to compute the alternatives in the next paragraph). For each alternate route r_i, if r and r_i share the first k ASes but differ at the $(k+1)$'th AS, then the k'th AS is said to prefer the $(k+1)'th$ AS on r over the $(k+1)'th$ AS on r_i. Each alternative route in the set $r_1, \ldots, r_m$ similarly yields a preference.

I generate the set of alternative routes by leveraging the path composition technique and applying it to the set of observed routes. Given an observed path from a source to a destination, I ignore the observed path and generate alternative routes by composing path segments. Comparing the actual route with the alternative routes that are of the same AS length yields a count on the number of times I observed AS_i preferring a route through a neighbor AS_j over a route through AS_k.

I store the preferences obtained above as 3-tuples (AS1, AS2 > AS3), where AS1 prefers a route through AS2 over a route through AS3 when both routes are of the same length. In Figure 5.4, the path $1-2-3-4$ is selected over the path $1-5-3-4$ because of a preference (1, 2 > 5). In some cases, I observe both 3-tuples (AS1, AS2 > AS3) and (AS1, AS3 > AS2). So, I include the preference (AS1, AS2 > AS3) only if it was observed at least three times as often as the preference (AS1, AS3 > AS2). If not, I ignore both preferences; I conjecture that such wavering preferences are likely due to load balancing by AS1.

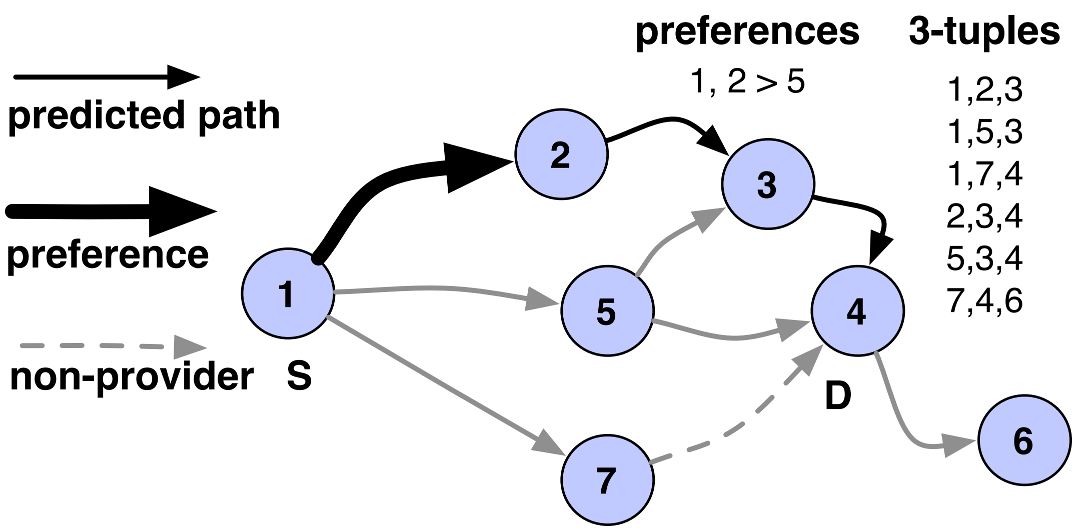


Figure 5.4: Predicting the path from S to D. Thicker lines show preferences, dotted lines show non-provider links, and dark lines show the prediction. The path $1-5-4$ cannot be chosen because the 3-tuple does not appear and $1-7-4$ cannot be chosen because 7 is not a provider for 4. $1-2-3-4$ is predicted because of 1's preference for 2 over 5.

AS preferences could also be independent of AS path length, i.e., an AS might prefer a path through one of its neighbors over another even if the path through the former is longer. My path prediction algorithm, which emulates Dijkstra's shortest path algorithm, cannot model such preferences. Incorporating such AS preferences into the model would necessitate a prediction algorithm that closely tracks BGP. Not only would such an algorithm incur a longer runtime to predict routes, because unlike the shortest path algorithm every edge might be traversed multiple times, but also my evaluation in Section 5.8 shows that my model seldom predicts an incorrect AS path length.

5.7 Incorporating traffic engineering

In many cases, I observe an edge from AS1 to AS2 on some route in the atlas, but never see this edge on a route terminating at AS2, i.e., when the destination is in AS2. In other words, the path segment (AS1, AS2, AS3) is seen on routes but never does a route end with the segment (AS1, AS2). The optimizations described above, the 3-tuple check and AS preferences, are insufficient to handle such cases.

To address this problem, I explicitly maintain information about provider ASes. For each AS, I determine its upstream neighbor ASes, i.e., the set of ASes observed immediately prior to this AS in the atlas. I also determine the set of providers for each AS, i.e., the set of ASes observed upstream of this AS when it is the origin. For the latter, I use both AS paths corresponding to the observed paths measured from my vantage points as well as BGP snapshots provided by RouteViews [53], RIPE [70], GEANT [36], and Abilene [90]. For 1,352 ASes out of a total of 27,515 ASes in the atlas, I find the set of providers to be a proper subset of the set of upstream neighbors. In these cases, the previous algorithms could give the wrong path. I refine the approach further to determine the provider set and upstream neighbor set on a per-prefix basis. In Figure 5.4, the path $1 - 7 - 4$ cannot be selected, even though it is shorter, because 7 is not a provider for 4.

I store the above information as follows. For ASes with identical provider and upstream neighbor sets, nothing needs to be stored. For each AS that has the same set of providers for all of its prefixes, the set of providers for the AS is explicitly stored. For each AS for which the set of providers varies across its originated prefixes, I partition the prefixes into disjoint sets such that the prefixes within a partition have the same provider set and explicitly store this provider set.

5.8 Evaluation

In this section, I evaluate the accuracy of the graph-based path prediction algorithm and study the contribution that each of the algorithm's components makes towards its predictive ability. I also quantify the storage requirements of the graph-based atlas and the other data required by the prediction algorithm, the stationarity of this data across days, and how the atlas would grow with additional vantage points.

5.8.1 Measurement data

I again leverage PlanetLab nodes as vantage points for gathering the atlas. The atlas I use in this evaluation comprises traceroutes from 197 PlanetLab nodes to one destination

Table 5.2: Breakdown of sizes of different components of graph-based atlas.

Dataset	Number of entries	Size on disk (in MB)
Inter-IP links	905K	3.27
IP to cluster mapping	234K	0.61
Prefix to AS mapping	247K	1.44
Inter-cluster links	309K	2.69
AS three-tuples	2.15M	1.76
AS preferences	9K	0.03
Provider mappings	20K	0.60
Total		**10.40**

each in 142K prefixes. All of these traceroutes were gathered over the course of a day. 234K distinct IP addresses are present in the atlas, with 905K links between them. These addresses and links map to 84K clusters with 309K inter-cluster links. The dataset obtained by combining these inter-IP links with inter-cluster links annotated with latencies and loss rates (I deal with measuring properties of inter-cluster links in Chapter 6), observed AS 3-tuples, inferred AS preferences, and the mapping of ASes to their providers is roughly 10MB in size. Table 5.2 shows the size associated with each of these components of the dataset; the inter-IP links constitute most of this data.

From the 197 vantage points, I choose a subset of 37 at random as representative end-hosts. I pick 100 random traceroutes performed from each of them. After discarding paths that did not reach the destination or have AS-level loops, I am left with a validation set of 2816 paths. To predict the paths from one of the 37 sources, I include all traceroutes from the remaining 196 vantage points in the TO_DST plane and 100 other randomly chosen traceroutes from this source in the FROM_SRC plane.

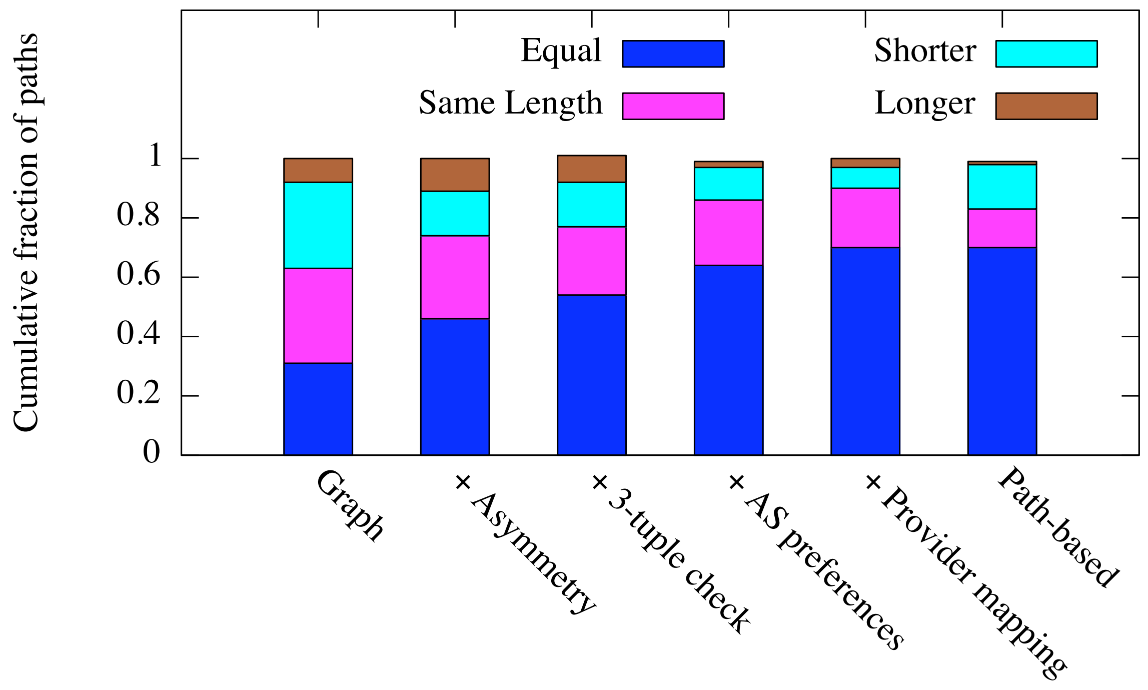


Figure 5.5: AS path prediction accuracy for measured traces as components are incorporated into the path prediction algorithm. GRAPH is the algorithm described in Section 5.2, and path-based is the path composition algorithm without the three-tuple check.

5.8.2 Can the algorithm predict AS paths accurately?

I evaluate the accuracy of the graph-based algorithm's ability to predict AS paths in my validation set. Figure 5.5 shows the improvement in accuracy of AS path prediction as each optimization is incorporated into the GRAPH algorithm. The fraction of paths for which I predict the AS path exactly right increases from 31% with GRAPH to 70% with all components of the algorithm included. Each of the four techniques added to GRAPH significantly improves its ability to predict paths. In fact, my final predictive model achieves the same AS path accuracy as the path composition technique without the three-tuple check, which uses a path-based dataset two orders of magnitude larger than the link-based atlas. Furthermore, the graph-based path prediction outdoes path composition in this setting in the ability to predict AS path length.

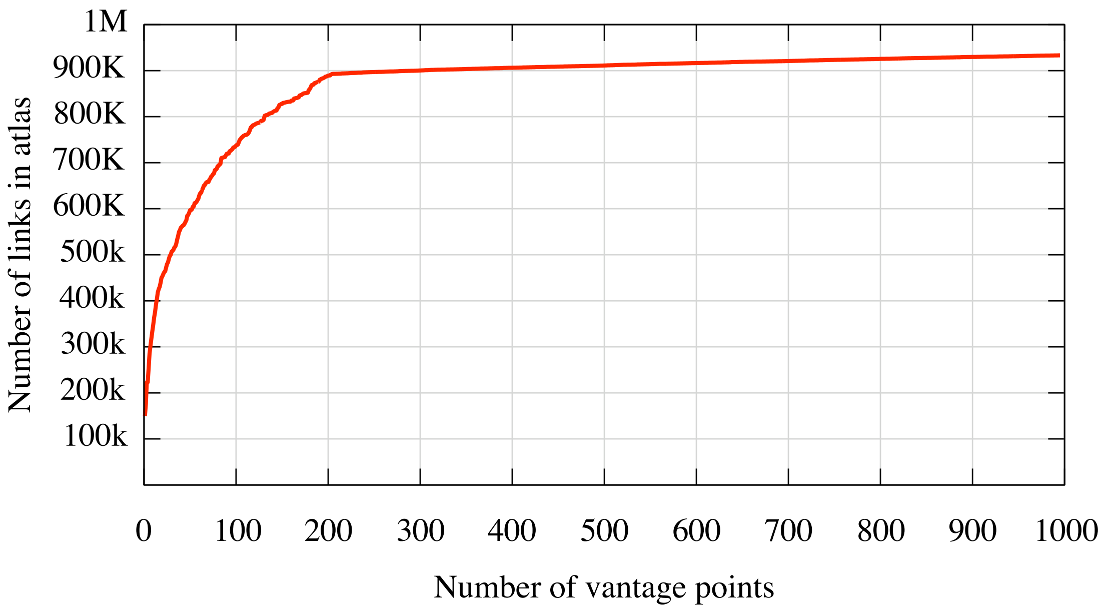


Figure 5.6: Increase in size of Internet atlas with addition of traceroute measurements from end-hosts. The core set of vantage points (PlanetLab nodes) issue traceroutes to destinations in 140K prefixes each. One end-host in each of the 100K edge prefixes is assumed to contribute 100 traceroutes.

As in Chapter 4, I leave the evaluation of path prediction accuracy at cluster-level for future work.

5.8.3 Does the atlas scale with end-host measurements?

Path prediction relies on measurements from end-hosts for improved prediction accuracy. This is borne out by the improvement in path prediction accuracy seen in Figure 5.5 when routing asymmetry is accounted for in the prediction algorithm. However, adding in more measurements could significantly inflate the size of the atlas. This would put into question the basic tenet of path prediction using an atlas of links—is the atlas still tractable if it includes end-host measurements?

To study this question, I use the DIMES measurement infrastructure [75]. The DIMES project runs an Internet measurement agent on a few thousand end-hosts distributed worldwide. I issued traceroutes from 845 DIMES agents to 100 randomly chosen destinations each over the course of a week in August 2007. This is representative of how we envision end-hosts integrating themselves into the information plane—end-hosts in each prefix contribute a few traceroutes to augment the comprehensive measurements gathered from PlanetLab.

As stated previously, measurements from PlanetLab find approximately 900K links. Figure 5.6 plots the number of inter-IP links in the atlas as measurements from new vantage points are incorporated into it. The first 197 vantage points in the figure are the PlanetLab nodes I use to build the atlas and DIMES agents account for the remaining. Including the measurements from the 845 DIMES agents into the atlas added approximately 30K links in total. Extrapolating, including traceroutes from end-hosts in all 100K prefixes at the Internet's edge would increase the number of links in the atlas from 900K to approximately 4M, a five-fold increase.

5.8.4 How stationary is the atlas over time?

My goal in developing an algorithm for path prediction on a graph-based atlas of the Internet is to enable distribution of a map of the Internet to end-hosts so that they can make performance predictions. Since routing in the Internet is not static, end-hosts would need to refresh their local atlas periodically.

To evaluate how often this refresh is necessary, I generated a daily atlas for 15 successive days. I then compared the atlas for each day during this period with that gathered on the first day. Figure 5.7 shows the difference in the set of inter-IP links observed across days; as seen in Table 5.2, other components of the atlas—the set of AS 3-tuples, AS preferences, and provider AS mappings—are significantly smaller and are stationary over long periods. The difference between a pair of atlases is computed as the union of the set of links removed from the first atlas and the set of new links added to the second one. I did not measure the stability of link properties.

A 20% change is observed between the atlases of the first and second day, but the flux in atlases observed thereafter is lesser. I believe the change in atlases between successive days is predominantly due to load-balancing within ASes. This is borne out by the fact

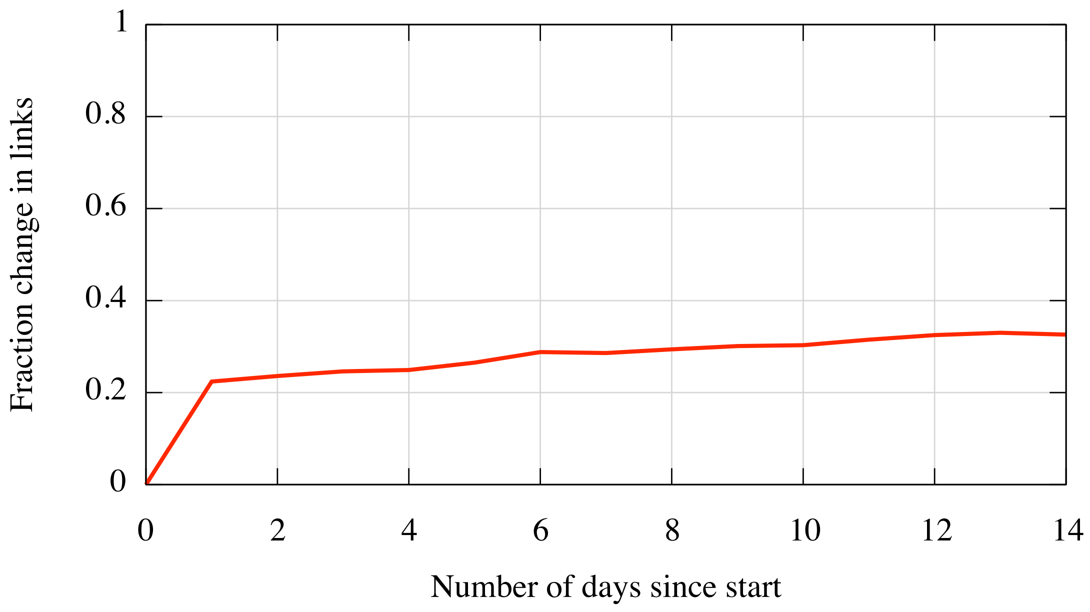


Figure 5.7: Fractional increase in number of inter-IP links on i^th day compared to the atlas on the first day.

that approximately 90% of the new links observed between any pair of consecutive days is constituted by intra-AS links.

5.8.5 Improved Path Composition Technique

The techniques discussed in this chapter are also applicable to the path prediction approach that works by composing path segments. I incorporate these techniques into the path composition algorithm to improve the accuracy of prediction using an atlas of paths. When two path segments are being spliced together, I check whether the sequence of ASes prior to, at, and after the point of intersection exists in the database of 3-tuples. I also ensure that the AS preferences detected are enforced when multiple candidate intersections pass the 3-tuple check. Figure 5.8 shows that these modifications to the path composition technique increase its ability to predict AS paths from 70% to 81%.

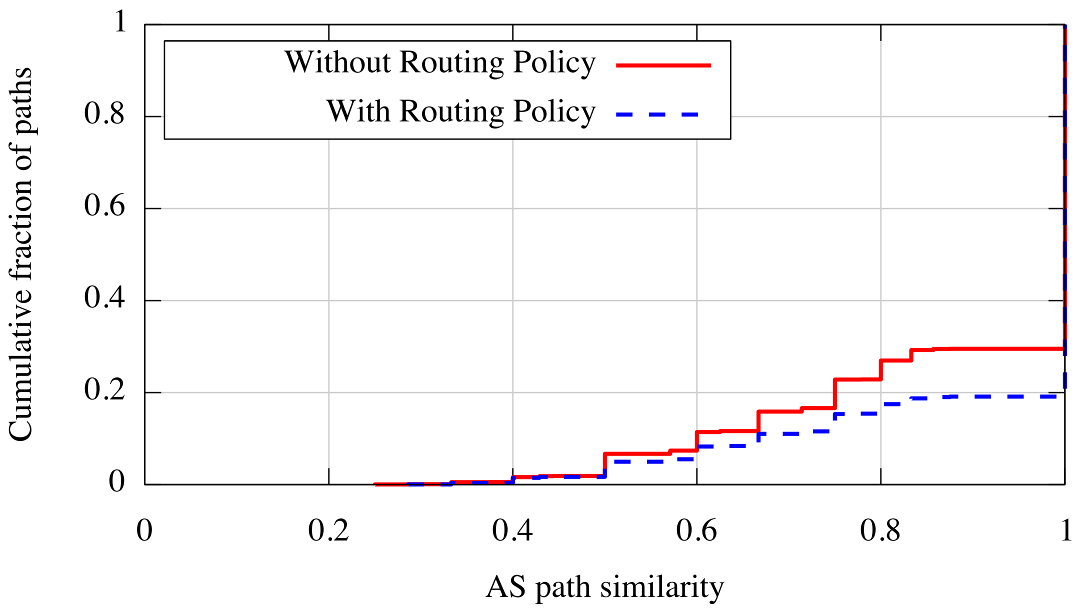


Figure 5.8: AS path prediction accuracy of the path composition technique with the *Min AS path, Min RTT* policy after incorporating inferred routing policies for ASes.

5.9 Summary

Path prediction by composition of path segments requires as input an atlas of measured paths, whose size will blow up as measurements are contributed by more vantage points. The size of a path-based atlas is proportional to the number of vantage points times the number of destinations probed times the average path length. With my current set of vantage points and destinations, the size of such an atlas is over 1GB after compression. As more vantage points contribute measurements, path prediction accuracy will increase, but at the cost of blowing up the size of its atlas.

Instead, in this chapter, I developed a model of path prediction that operates on an atlas of Internet links. The prediction algorithm that implements commonly known facets of Internet routing—shortest AS path, valley-free AS paths, and early exit routing—predicts AS paths with only 30% accuracy. To improve the prediction accuracy, I augment the algorithm and its input with four optimizations—1) I account for routing asymmetry by differential treatment of links observed on paths measured from end-hosts as opposed to those observed on paths from vantage points to all destinations, 2) I account for observed routing behavior by storing all sequences of three successive ASes seen on measured paths, 3) I detect every AS's preferences amongst its neighbors, and 4) I store for every AS, the subset of its upstream neighbors that serve as its providers. Incorporation of these changes significantly improves the AS path prediction accuracy obtained with the graph-based algorithm.

Chapter 6
MEASURING LINK METRICS

The communication between a pair of end-hosts on the Internet typically involves an exchange of packets between the hosts. Packets travel from the source to the destination, and responses travel back from the destination to the source. Thus, the quality of communication depends on the properties of the links traversed on the forward path through the Internet from the source to the destination and on the reverse path back. In Chapters 4 and 5, I described the algorithms I developed to predict the path between arbitrary end-hosts. In this chapter, I examine the remaining piece of the puzzle for predicting path performance—estimating properties of links.

6.1 Overview

The impact of intra-PoP links on path performance is minimal. Packets incur sub-millisecond latencies while traversing links between routers in the same PoP. Also, intra-PoP links are typically significantly over-provisioned compared to long distance links, and so packets are rarely dropped or subjected to queueing delays on such links. Therefore, I focus on measuring the properties of inter-PoP links. I use the algorithms described previously in Section 4.5 to cluster together router interfaces in the measured topology that belong to the same PoP in an AS and measure the properties of inter-cluster links thus discovered. Table 6.1 summarizes the various techniques employed to measure link properties such as latency, loss rate, and bandwidth capacity.

The latency inference techniques I use when paths are predicted using the path composition approach and using the graph-based approach are different, and I describe these separately. I then outline the measurement techniques I employ to measure loss rates of links. The techniques for measuring latencies and loss rates of individual links are also applicable to path segments. I follow up with a description of the setup I use to orchestrate

Table 6.1: Summary of techniques employed to measure latency and loss rate of links and path segments.

Technique	**Description**	**Goal**	**Section**
Traceroutes from vantage points	Traceroutes launched for gathering topology information also provide latencies to intermediate hops	Measure latencies of path segments	Section 6.2
Record-route enabled probes	Perform record-route enabled traceroutes and compare first hop on reverse path to last-but-one hop on forward path	Detect symmetry, Measure link latency	Section 6.3.1
UDP probes	Send UDP probes and compare the source address of the response with the address probed	Detect symmetry, Measure link latency	Section 6.3.1
IP timestamp probes	Send ICMP probes with IP timestamp enabled to detect the presence of an interface on the reverse path	Detect symmetry, Measure link latency	Section 6.3.1
Pings between vantage points	Issue pings between all vantage points, log timestamps at the source and destination of every ping, and compare the forward and reverse one-way delays	Synchronize clocks of vantage points	Section 6.3.2
Spoofed probes	Probe interface with source address set to another vantage point which has a symmetric path to the interface	Measure one-way path latency	Section 6.3.2
Large ICMP probes	Send 100 1KB ICMP probes from vantage points to an interface	Measure one-way path loss rate	Section 6.4.1
Frontier search	Perform breadth-first search on measured topology from all vantage points	Distribute link measurements	Section 6.4.2

the measurement of link properties from my vantage points. I conclude this chapter with an evaluation of my ability to estimate path properties using the measured link properties and the path prediction techniques described in previous chapters.

6.2 Measuring path segment latency

Recall that the path composition approach for predicting the route from a source S to a destination D works by splicing a path segment $S.I$ with a path segment $I.D$. To estimate the *one-way* latency of $S.I.D$, i.e., the latency that a packet would incur while traversing the path, I estimate the latency along the segments $S.I$ and $I.D$ and sum them up.

The path segment $I.D$ is considered in the path composition technique only when it has been observed on a traceroute from some vantage point V. Thus, the traceroute from V that contains the segment $I.D$ would include RTTs from V to I and D. Using these RTT measurements, I estimate the latency from I to D as $RTT(V, D)/2 - RTT(V, I)/2$. Note that this implicitly assumes that the path from V to D is symmetric (or at least the portion of the path that spans from I to D), which is not always true. To account for the error introduced by path asymmetry into the latency estimate for a path segment $I.D$, I leverage the observation that the same path segment may have been observed on traceroutes from multiple vantage points to D. Some of these traceroutes may correspond to symmetric paths, and others may not. Every traceroute on which the path segment was observed yields a sample based on the formula above for the segment's latency. I estimate the latency of the path segment as the median of these samples so as to discard outliers introduced due to path asymmetry.

Similarly, the path segment $S.I$ is observed either on a traceroute from a vantage point to a passive client, or on a traceroute from an active client. The former case is identical to that of estimating the latency of the segment $I.D$ as above. In the latter case, each traceroute from S on which the segment $S.I$ was observed yields a latency sample $RTT(S, I)/2$, and I consider the median of such samples as the estimate for the segment's latency. Even with access to traceroutes from the source of the path segment $S.I$, it is hard to estimate the one-way latency along the segment because traceroute records only round-trip measurements of latency.

Note that the output of traceroute contains RTTs to each router interface on a path, but the path composition technique identifies intersections at the granularity of clusters. Hence, when a measured route traverses multiple router interfaces in the same cluster, I use the RTT to the cluster to be the minimum of the RTTs measured to the interfaces in this cluster. Since routers in the same cluster are typically geographically co-located, RTTs to interfaces in a cluster from the same vantage point are usually similar.

6.3 Measuring link latency

Predicting route metrics, especially latency, is much harder with a graph representation of the atlas, as compared to storing the original set of measured routes. The graph-based approach for path prediction maintains an atlas of inter-IP links. In this case, I estimate the latency along a predicted path by summing up the latencies of all the links on the path. To enable this, I store latency estimates for the corresponding inter-cluster links—several inter-IP links map to the same inter-cluster link. In this section, I focus on performing measurements to accurately determine latencies of inter-cluster links.

Unfortunately, because of the high variance seen across the latency estimates for a link when it is observed along multiple traceroutes, the error introduced by adding up latency estimates for individual links is significantly higher than that obtained by adding latency estimates for longer segments. As I later show in Section 6.5, 40% of links display a variance of more than 20ms, with several links showing a variance greater 100ms!

I apply the following heuristic to reject outliers: if a link (x, y) occurs on a traceroute from s, and the RTT from s to y is t, then the latency of link (x, y) can be at most t. If a traceroute from another source or to another destination traverses the same link and yields a link latency estimate greater than t, I discard that sample when computing the median latency for the link. Although the constraint improves prediction accuracy, the absolute error is still high, making the case for more accurate measurement of link latencies.

To enable accurate measurement of link metrics, I supplement the above heuristic with a two pronged approach. First, I use several complementary techniques to identify symmetric paths—paths for which I can more easily get accurate data using round-trip measurements. Second, I measure the latencies of other links that do not appear along symmetric routes,

```
SYMM_TRACE($P, x$):

1. Perform traceroute and record-route enabled pings from $P$ to $x$.
2. If $x$ is within 8 hops of $P$, obtain the the first hop on the reverse route from $x$ to $P$
   using the IP record-route option (usable up to 9 hops).
3. If the first hop on the reverse route is in the same cluster as the last-but-one hop on
   the forward route, declare the last link to be traversed symmetrically on the route to
   $x$.
```

Figure 6.1: Procedure to identify symmetry using traceroute and IP record-route.

```
SYMM_UDP($P, x$):

1. Send a probe from $P$ to $x$ such that its TTL expires at $x$. The ICMP response will
   contain the outgoing interface $d$ on the reverse route.
2. If $d$ is the same as the last interface observed on the traceroute from $P$ to $x$, we know
   that the last link is symmetric
```

Figure 6.2: Procedure to identify symmetry using TTL-limited UDP probes.

by leveraging measurements of symmetric paths.

6.3.1 Identifying symmetric links in routes

I combine three separate novel procedures to identify which links are traversed symmetrically. The procedures each rely on a different type of probe: 1) record-route enabled ICMP probes, 2) UDP probes, and 3) timestamp enabled probes. Each of the three works only in some cases, due to protocol and router configuration limits. However I show that, in combination, they serve as a useful toolkit to identify symmetric links.

Record-route ICMP Probes: An ICMP probe with the record route IP option enabled records the interfaces encountered by the probe along its forward and reverse paths. IP limits the number of recorded interfaces to be 9. So as long as the far end of a probed link is within 8 hops of a vantage point, I can use the technique shown in Figure 6.1 to determine

```
SYMM_TIMESTAMP($P, x$):
  1. Let $d$ be the last-but-one hop from $P$ to $x$. Send a probe from $P$ to $x$ with the IP
     timestamp prespec option enabled for $d$, $x$, and $d$ in that order.
  2. If all three timestamp fields are filled, we know that $d$ lies on the forward route from $P$
     to $x$ because $d$ filled in its timestamp before $x$. We also know that $d$ is on the reverse
     route from $x$ to $P$ since $d$ filled in its timestamp after $x$, thus implying that the link
     between $d$ and $x$ is traversed in a symmetric manner.
```

Figure 6.3: Procedure to identify symmetry using the IP timestamp option.

whether the link is traversed in a symmetric way.

The caveat of this technique is that from each vantage point, only a fraction of clusters will be reachable within record route's limit of 9 hops.

UDP Probes: A router responding to UDP probes fills in the IP address of the outgoing interface in its response. On the other hand, when a router receives a TTL-expired message, e.g., as part of a traceroute, it fills in the IP address of the incoming interface in its response. I can use these pieces of information in the technique shown in Figure 6.2 to identify when the first link on the reverse path is the same as the last link along the forward path.

This technique suffers from the caveat that many routers do not respect the requirement of responding to UDP probes with the outgoing interface. I restrict this technique to those routers that fill in different outgoing interface values for probes from different vantage points.

Probes to query router timestamps: IP allows probes to query the timestamps of a set of specific routers along a path. I use timestamp query probes in the technique shown in Figure 6.3 to again identify symmetric traversal of links.

Many ISPs however not only configure their routers to ignore the IP timestamp option but also drop packets with this option turned on.

All of the above techniques identify links (x, y) that packets traverse in a symmetric manner. For symmetric links, I estimate the latency of (x, y) as half the difference of the round trip delays $RTT(P, x)$ and $RTT(P, y)$ along the route to D. Whenever I identify a particular link as symmetric in at least one route, I consider only identifiably symmetric

```
CLOCK_DIFF($A, B$):

  1. Ping $B$ from $A$, and receive the reply to measure the RTT $d$.

         $d = d_1 + d_2 + c_3 - c_2$                                        (6.1)

  2. Let $c_1$ be the value of the clock on $A$ when the ping probe was sent out and $c_2$ be the
     value of the clock on $B$ when the probe is received.

         $c_2 = c_1 + d_1 + \delta$                                         (6.2)

  3. Let $c_3$ be the value of the clock on $B$ when the ping-reply was sent out and $c_4$ be the
     value of the clock on $A$ when the ping-reply is received.

         $c_4 = c_3 + d_2 - \delta$                                         (6.3)

  4. By the definition of RTT, we have:

         $d = c_4 - c_1$                                                    (6.4)

  5. If equations 6.1, 6.2, 6.3, and 6.4 remain consistent after adding the constraint $d_1 = d_2$,
     compute the corresponding value of $\delta$. Else, return $\perp$.
```

Figure 6.4: Procedure to measure clock difference between vantage points A and B.

measurements of that link (which may be from different vantage points) when calculating the median to assign as the link latency. The techniques complement each other in that a link is symmetric if any of the techniques is successful. A path segment is symmetric if every link along it is symmetric.

6.3.2 Measuring link latencies using spoofed probes

I now propose a distributed technique to measure link latencies, as well as one-way delays of Internet routes, even for asymmetric links. To do so, I use measurements to synchronize clocks across the vantage points, issue probes with spoofed source address values, and estimate one way delays by taking into account measurements of known symmetric paths.

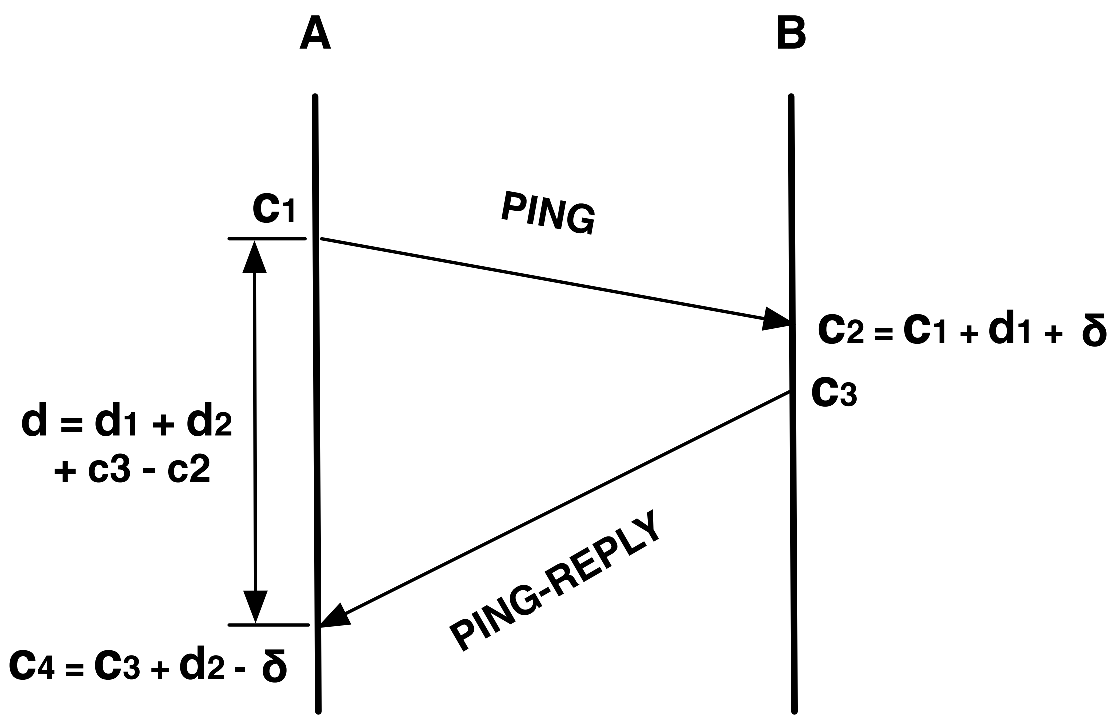


Figure 6.5: Computing the clock difference between nodes A and B connected by a symmetric path. δ is the clock skew between A and B. If $d_1 = d_2$ and all the equations remain consistent, $\delta = \frac{c_2 - c_1 + c_3 - c_4}{2}$.

I first outline how I synchronize clocks across vantage points and then describe how I use spoofed measurements to determine one-way delays.

Synchronizing clocks

Let A and B be two end-hosts whose clocks need to be synchronized. Let the clock difference between them be δ. Let the one-way delay from A to B be d_1 and that from B to A be d_2.

The procedure in Figure 6.4 exactly computes δ when feasible. The timing diagram in Figure 6.5 illustrates the algorithm. Although equations 6.1, 6.2, 6.3, and 6.4 in Figure 6.4 have three unknowns d_1, d_2, and δ, they are not linearly independent. However, if the equations remain consistent with the additional constraint $d_1 = d_2$, then I can solve for δ and know that d_1 was indeed equal to d_2. On the other hand, if d_1 were not equal to

d_2, then the equations become inconsistent and I can not compute δ. Thus, if the one-way delays in either direction are equal, CLOCK_DIFF infers as much and also computes the clock difference, using just one ping! To minimize error in δ, I repeat CLOCK_DIFF five times and use the value returned only if multiple trials return values within 5 ms of each other. Since CLOCK_DIFF does not depend on router support, its implicit determination of symmetry along a path works much more often than explicitly trying to determine symmetry using the techniques described earlier.

By iteratively applying CLOCK_DIFF, one can compute the clock difference between pairs of vantage points even if they have an asymmetric route. Let C and D be one such pair. If there exists some vantage point A such that both A, C and A, D have symmetric forward and reverse path delays, then it is straightforward to compute the clock difference between all three nodes. My use of clock skew only if consistent across multiple trials minimizes propagation of error across nodes. Of the 203 PlanetLab sites I used in a sample run of clock synchronization, CLOCK_DIFF helped synchronize clocks between 177 sites.

Spoofing technique to determine one-way latency

Finally, I need to determine the one way latency along routes. Let (x, y) be a link observed from some vantage point P. The latency of (x, y) is the difference between the one-way delays $L(P, x)$ and $L(P, y)$. Figure 6.6 gives the procedure to measure (P, x) by first synchronizing clocks on the vantage points and then leveraging round-trip measurements of previously identified symmetric paths. The technique is also illustrated in Figure 6.7.

Note that the technique outlined above handles the general case where the vantage point P' is not capable of issuing spoofed probes. This generalization is motivated by the fact that only a few vantage points can issue spoofed probes. For example, some of the nodes on the RON testbed permit spoofing, while none of the PlanetLab nodes do so.

6.3.3 Putting latency samples together

For each link, I combine latency samples from multiple sources as follows. First, I consider the latency samples obtained from symmetric traversals of the link. If there are multiple

LATENCY(P, x):

1. Synchronize clocks between as many vantage points as is feasible (§ 6.3.2).
2. Identify a vantage point P' that has a symmetric route to x (§ 6.3.1). Measure the one-way delay $L(x, P')$ as half of $RTT(P', x)$.
3. Pick a vantage point R from which spoofed probes can be sent and that is synchronized with P'. Send a probe from R to x with the source spoofed as P' to measure $L(R, P') = L(R, x) + L(x, P')$. Subtract the known value of $L(x, P')$ to obtain $L(R, x)$.
4. Send a probe from R to x with the source address spoofed as P to measure $L(R, P) = L(R, x) + L(x, P)$. Subtract the known value of $L(R, x)$ to obtain $L(x, P)$.
5. Measure $RTT(P, x)$. Subtract the known value of $L(x, P)$ to obtain $L(P, x) = RTT(P, x) + L(R, P') - L(R, P) - \frac{RTT(P',x)}{2}$.

Figure 6.6: Procedure to measure one-way latency from a vantage point P to a router interface x.

paths on which the link was traversed in a symmetric manner, I consider the median of the link latency estimates obtained from all of such paths as the link latency. In the absence of any symmetric traversals of the link, I consider the link's latency to be that measured using spoofed probes. Finally, if the link's latency could not be measured using spoofed probes, e.g., because vantage points with symmetric paths to either end of the link could not be identified, I fall back to the median of the latency samples obtained from traceroutes, throwing out samples that violate the heuristic bound.

6.4 Measurement of other link attributes

I next outline the details of how loss rate is measured. Previous research has proposed several ways to measure each of these properties; my goal is not to evaluate the various approaches but rather to integrate these techniques into a useful prediction system. These techniques are amenable to replacement as the state-of-the-art is advanced by the Internet measurement community.

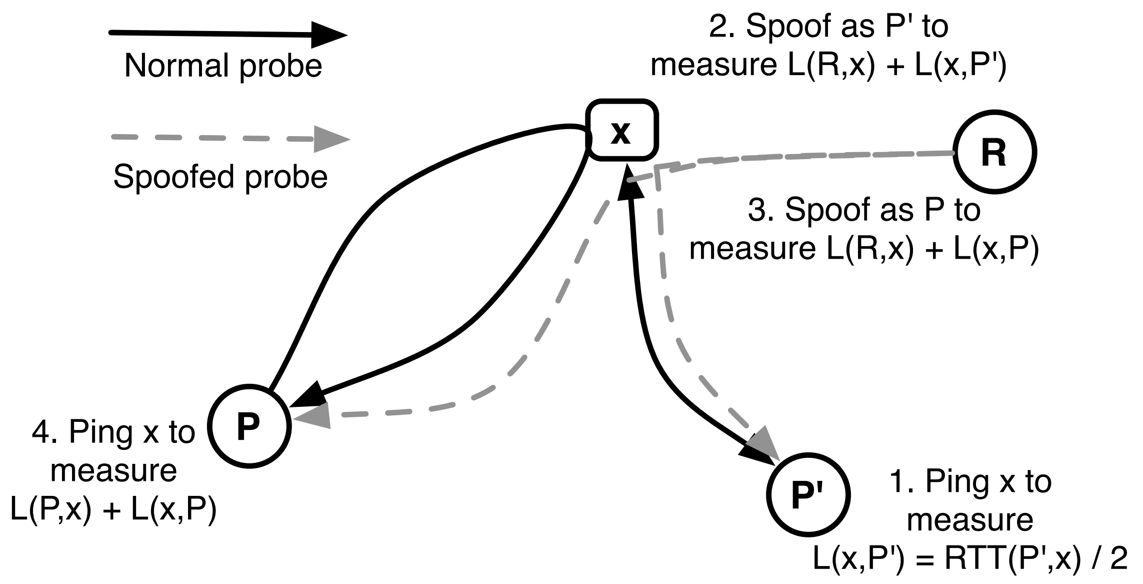

Normal probe
Spoofed probe
2. Spoof as P' to measure L(R,x) + L(x,P')
x
R
3. Spoof as P to measure L(R,x) + L(x,P)
P
4. Ping x to measure L(P,x) + L(x,P)
P'
1. Ping x to measure L(x,P') = RTT(P',x) / 2

Figure 6.7: Spoofing technique to measure one-way latency from P to x. P' is known to have a symmetric path to x, and hence, $L(P', x) = L(x, P')$. R is able to send source-spoofed probes.

6.4.1 Measuring Loss Rate

Any link in my atlas is one observed on the path measured from one of my vantage points to some destination. In my measurement of link loss rates, I assume that packet loss is independent across links. Therefore, the key challenge to inferring the loss rate on a particular link is to measure loss rates along the forward paths from the vantage point from which the link was observed to routers at either end of the link. These one-way loss rates can then be used to infer the link loss rate.

To determine the loss rate of a link (x, y) observed on the path from vantage point P to destination D, I measure the one-way loss rates along the path segments (P, x) and (P, y). I then estimate the loss rate that can be attributed to link (x, y) using the fact that a packet is not dropped on the path segment (P, y) if the packet is dropped on neither (P, x) nor (x, y), i.e., $(1 - loss(P, y)) = (1 - loss(P, x)) \cdot (1 - loss(x, y))$.

I perform loss rate measurements along path segments from vantage points to routers in the core by sending out probes and determining the fraction of probes for which I get responses. I use the simple method of sending TTL-limited singleton ICMP probes with a 1000-byte payload. When the probe's TTL value expires at the target router, it responds with a ICMP error message, typically with a small payload. When a response is not received, one cannot determine whether the probe or the response was lost, but there is some evidence from previous studies that small packets are more likely to be preserved even when routers are congested [49]. I therefore currently attribute all of the packet loss to the forward path; the development of more accurate techniques is part of future work. To account for noise in measurements, I measure the loss rate along a path segment by sending out 100 probes along the segment and recording how many are lost.

6.4.2 Orchestrating the Measurement Tasks

After topology measurements have been gathered from all vantage points, I operate on a compact routing topology, where each *node* in the topology is a cluster of interfaces and each *link* connects two clusters. I then seek to determine the latency and loss rate of each inter-cluster link that can be used to predict path performance. To achieve this goal, a centralized agent is used to distribute the measurement tasks such that each vantage point is assigned to repeatedly measure only a subset of the inter-cluster links. The centralized agent uses the compact routing topology to determine the assignments of measurement tasks to vantage points, communicates the assignment, and monitors the execution of the tasks.

There are three objectives to be satisfied in assigning measurement tasks to vantage points. First, I want to minimize the measurement load by measuring each link attribute from only a few vantage points (I employ more than one to correct for measurement noise). Second, the measurement should be load-balanced across all vantage points, *i.e.*, each vantage point should perform a number of measurements proportional to its capacity. Third, in order to measure the properties of each link as accurately as possible, links are preferably measured from the vantage point that is closest to it.

I have developed a "frontier" algorithm to perform the assignment of tasks to vantage points. The algorithm works by growing a frontier rooted at each vantage point and having each vantage point measure only those links that are at its frontier. The centralized agent performs a Breadth-First-Search (BFS) over the measured topology in parallel from each of the vantage points. Whenever a vantage point is taken up for consideration, the algorithm performs a single step of the BFS by following one of the traceroute paths originating at the vantage point. If it encounters a link whose measurement task has been assigned already to k other vantage points (k is a threshold chosen based on desired level of redundancy), it continues the BFS exploration until it finds a new link that has not been seen before. This process continues until all the link measurements have been assigned to some vantage point in the system. Improving the frontier algorithm to account for factors such as variation in access link bandwidth paths across vantage points and preferentially measuring a link along a symmetric path is future work.

The centralized agent uses the above algorithm to determine the assignment of tasks and then ships the tasklist to the respective vantage points. Each target link is identified by the traceroute path that the vantage point can use to reach the link and by its position within the traceroute path. If a vantage point is no longer capable of routing to the link due to route changes, the vantage point reports this back to the centralized agent. The centralized agent accounts for changes in routing by assigning every link to be measured by multiple vantage points.

6.5 Evaluation

In this section, I evaluate my estimates of link metrics and path properties. First, I evaluate the relative improvement in link latency estimates obtained using the proposed latency measurement techniques. Second, I use the measured link latencies and loss rates in combination with the path prediction techniques described in previous chapters to estimate end-to-end latency and loss rate, and evaluate the accuracy of these estimates. All of my validation is performed on paths between PlanetLab nodes; in the future client measurements, e.g., from DIMES [75], can be used to broaden the validation set. The number of PlanetLab nodes used varies with each experiment because of the variable availability of some nodes.

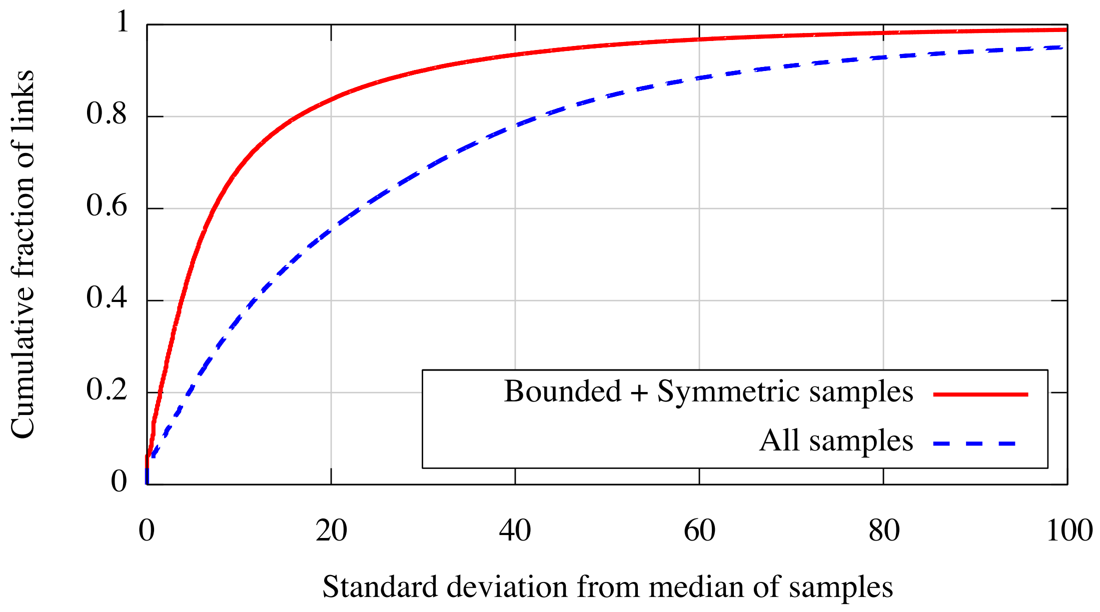


Figure 6.8: Spread in link latency samples when using my latency measurement techniques as opposed to using latency samples from traceroutes. Only links with multiple latency samples are considered, and the spread in samples is computed as the standard deviation from the median.

6.5.1 Accuracy of link latency estimates

First, I examine the relative improvement in link latency estimates brought about by my measurement techniques presented in Section 6.3. Figure 6.8 compares the spread in the latency samples that I consider for a link (those from the most preferred technique that applies) with the spread observed across all samples, plotting the standard deviation from the median of the samples. My measurement techniques (Section 6.3) yield multiple estimates for a link either when a link has multiple symmetric traversals or when I fall back on the median of samples that satisfy the heuristic bound. The latency samples obtained using my techniques display significantly less variance. The median standard deviation decreases from 16ms to 5ms.

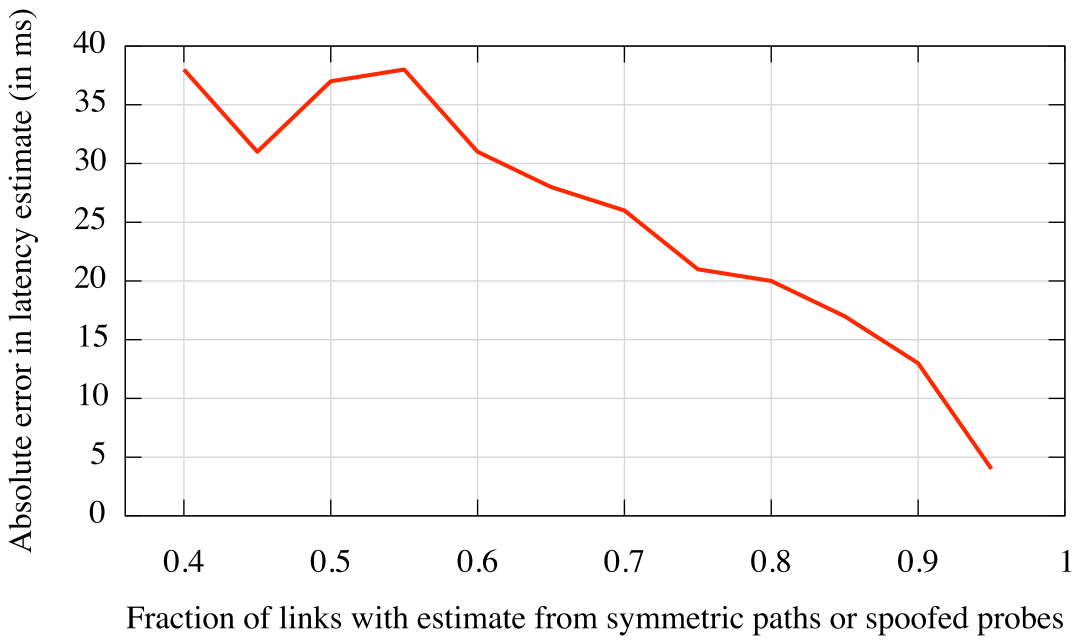


Figure 6.9: Accuracy of end-to-end latency estimates for known intra-PlanetLab paths with respect to the fraction of links with a symmetry- or spoofing-based latency estimate.

6.5.2 Accuracy of latency estimates for known paths

To evaluate the accuracy of my link latency estimates, I issued traceroutes from each PlanetLab node to every other node and considered those node pairs for which I was able to measure the route in both directions. I estimate the latency between each pair of nodes by summing up the latencies of links on both the forward and reverse routes. For each path, I compute the fraction of links for which I had a latency estimate using spoofing or symmetric traversal. I bin together all paths that have a similar value for this fraction. Figure 6.9 plots the 95^{th} percentile error in end-to-end latency estimates in each bin, using a bin size of 0.05. The more my techniques apply to a path, the better my end-to-end latency estimate. The accuracy of my latency predictions does not show any correlation with the actual latency, thus ruling out the alternative explanation that paths with a greater fraction of symmetric links are shorter.

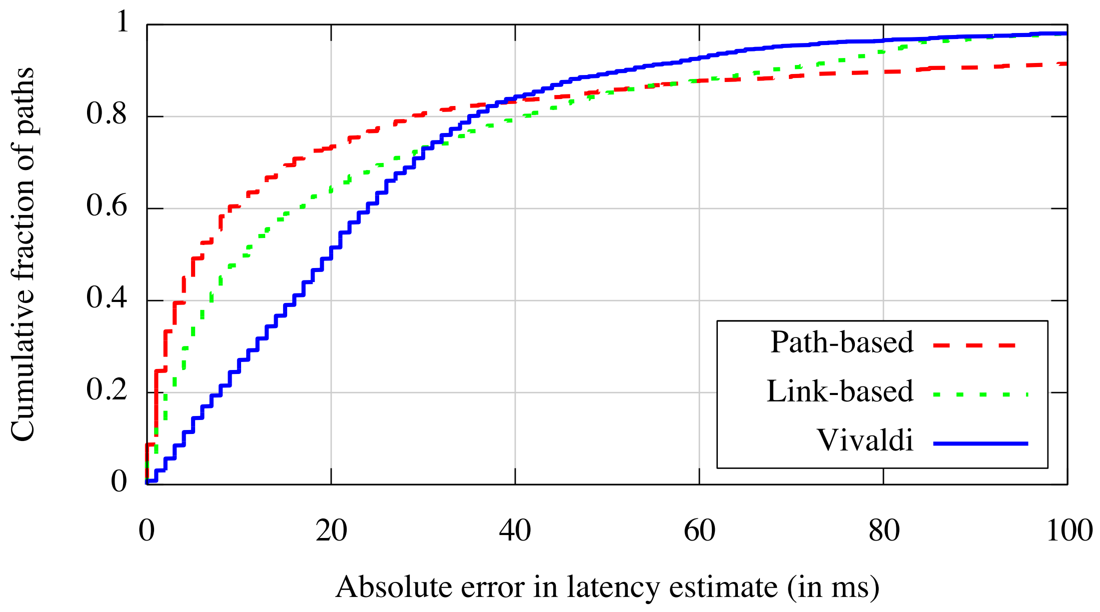


Figure 6.10: Accuracy of end-to-end latency estimates along paths to arbitrary destinations. Latency estimates when paths are predicted using both prediction techniques are compared with estimates obtained with Vivaldi.

6.5.3 Accuracy of latency estimates for predicted paths

Next, I evaluate my ability to estimate latencies along unknown paths, where techniques from earlier chapters are used to predict the route. With both path composition and graph-based path prediction strategy, I estimate the latency between a pair of end-hosts by predicting the forward and reverse routes between these hosts and summing up the latency estimates of links on these routes. While using the path composition technique described in Chapter 4 to predict paths, I estimate the latencies of path segments as outlined in Section 6.2. I then estimate the latency along a predicted path by summing up the latencies of the segments that are spliced together to produce that path. On the other hand, I estimate the latency along a path predicted using the graph-based technique (Chapter 5) by summing up my estimates for the inter-cluster links on the path. Incorporating link latencies from symmetric traversal of links into the path composition technique is future work.

I use the same measurement dataset as used in Section 5.8—paths from 37 PlanetLab nodes chosen at random to destinations in 100 random prefixes each—for evaluation of my path latency estimates. Figure 6.10 shows the errors in latency estimates obtained using either path prediction technique. The median latency estimation error with graph-based path prediction is 11ms. The path composition technique yields an even lower median error of 6ms, since estimates of latencies along path segments tend to be more accurate than those of individual links.

I also compare my latency estimation accuracy with that of one of the best existing coordinate-based systems, Vivaldi [18]. Feeding in the latencies of all paths in the measured atlas into Vivaldi, I generate 2-dimensional Euclidean coordinates with height vector for all end-hosts present in the evaluation dataset. I use these coordinates to estimate the latencies of all paths in the validation set. Figure 6.10 compares the latency estimates obtained using Vivaldi with the estimates obtained using my techniques. My latency estimates, both in terms of relative and absolute error, are often better than those yielded by Vivaldi's coordinates. For example, 73% of predictions obtained using the path composition approach are within 20 ms of actual latency, while only 49% of Vivaldi's are.

6.5.4 Accuracy of path loss rate estimates

I next consider how well I can predict loss rates. I measured the loss rate along each of the validation paths, and I also estimated the loss rate of each inter-cluster link in the atlas. I then estimate the loss rate along a path by composing the loss rates of the links along the predicted path. If the predicted path between a source s and a destination d goes through a sequence of clusters $c_1, c_2, \ldots, c_k$, then I estimate the end-to-end loss rate $L(s,d)$ using the link loss rates $L(c_1,c_2), L(c_2,c_3), \ldots, L(c_{k-1},c_k)$ as $(1 - L(s,d)) = (1 - L(c_1,c_2)) \cdot (1 - L(c_2,c_3)) ... (1 - L(c_{k-1},c_k))$. In other words, the probability a packet is not dropped on the path is equal to the product of the probabilities that the packet is not dropped on any of the links along the path. Figure 6.11 plots the accuracy of loss rate estimates obtained. Since coordinate systems, such as Vivaldi, can only estimate latency, I restrict my evaluation to estimates obtained using both of my approaches for path

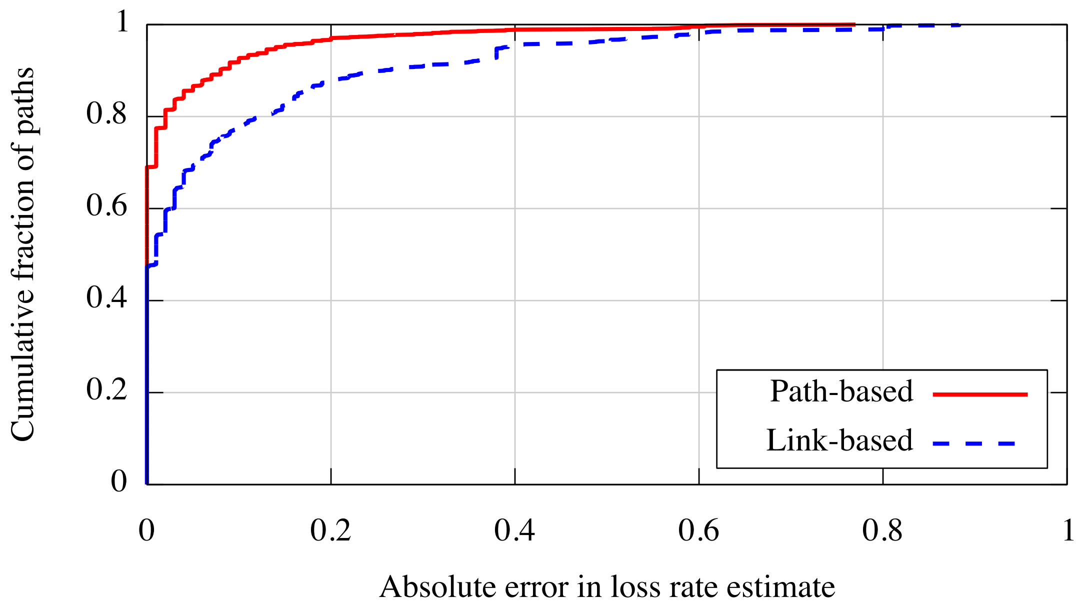


Figure 6.11: Accuracy of loss rate estimates along paths predicted using both prediction techniques to arbitrary destinations.

prediction. The loss rate estimates obtained using either approach are reasonably accurate for the dataset I examine—the fraction of paths with less than 5% error is more than 85% with the path composition approach, and more than 68% with the graph-based technique.

6.5.5 Accuracy of application-level metrics

Applications such as peer selection and detour routing benefit from the ability to discern which destinations have low latency from a source. I therefore also assess latency estimation from the perspective of ranking different destinations in terms of latency from a common source. To quantify each technique's predictive ability on this criterion, I use the following metric. From each source, I determine the 10 closest nodes in terms of actual measured RTT. I then do the same using estimated latencies and compute the intersection between the actual and predicted sets of 10 closest nodes. Figure 6.12 plots the cardinality of this intersection for each source in the validation set used in Section 5.8—paths from 37

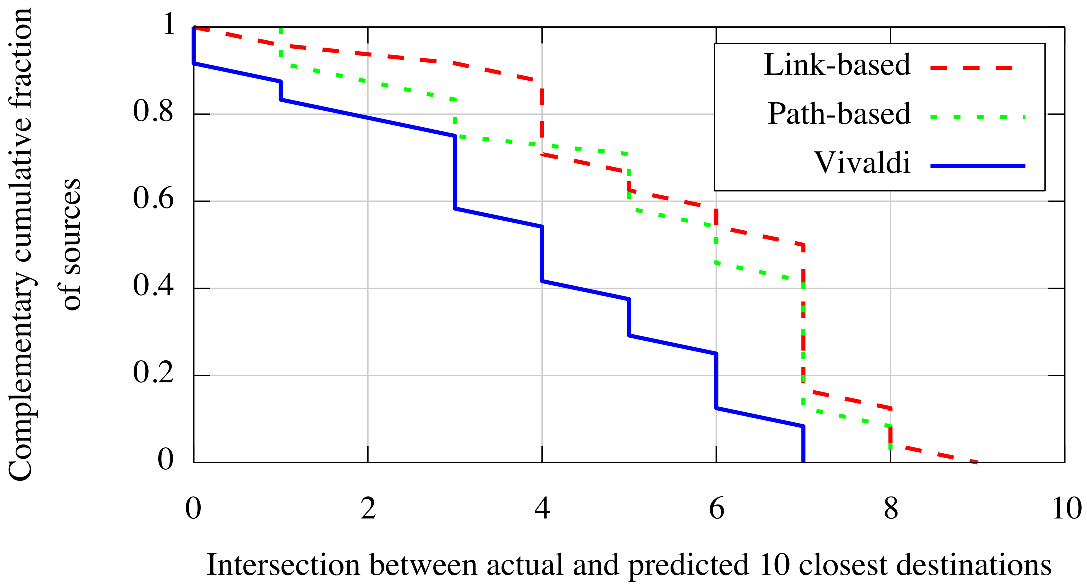
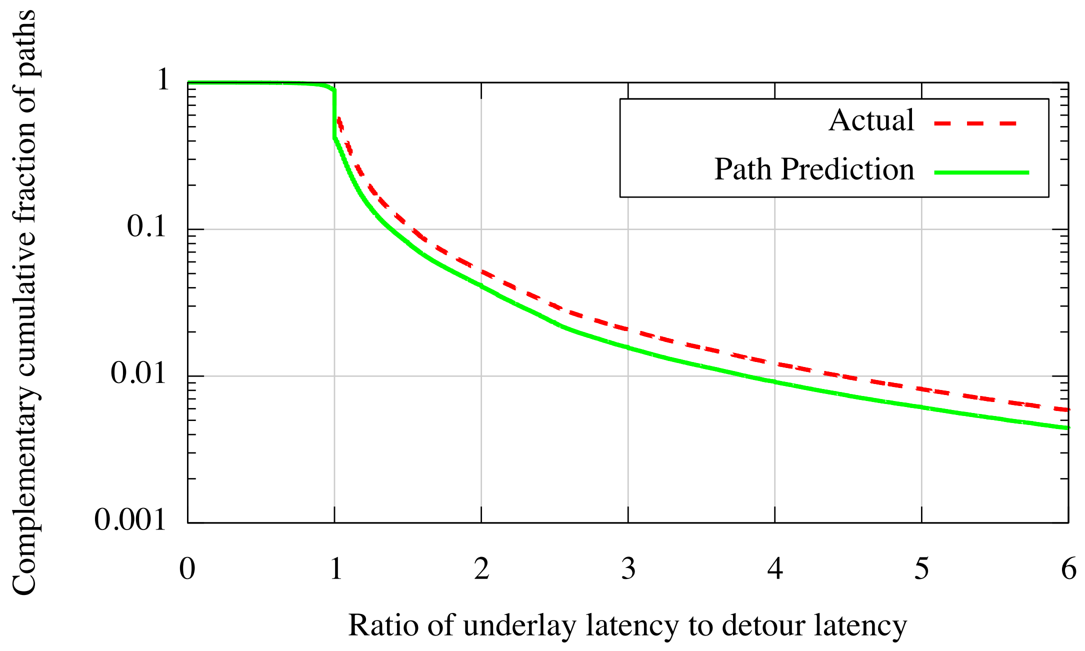


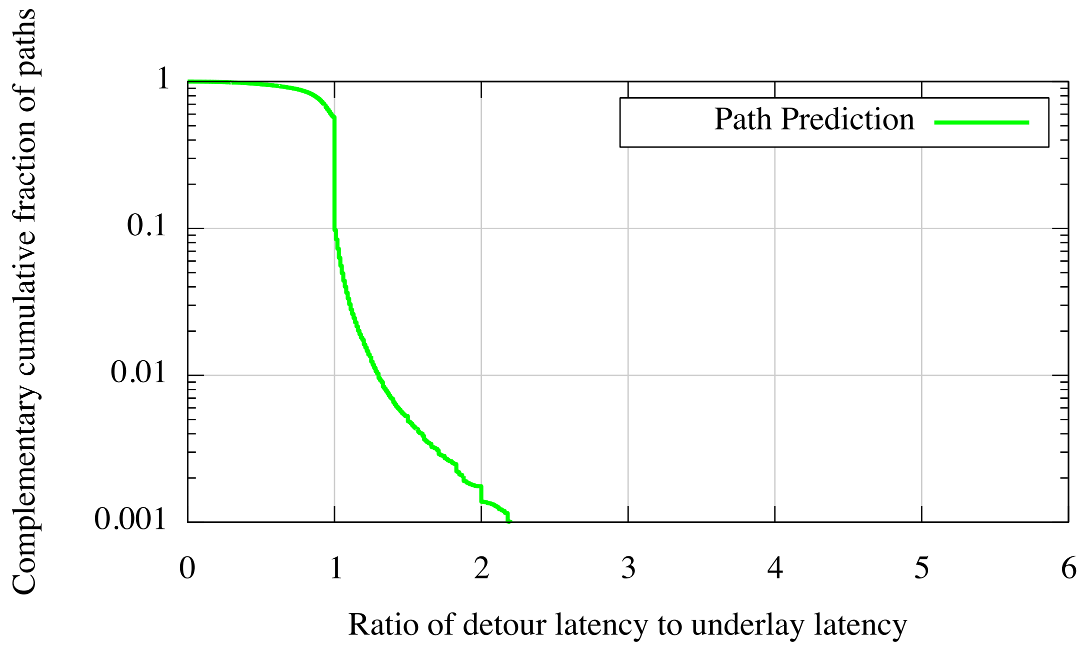
Figure 6.12: On the dataset comprising paths from 37 PlanetLab nodes chosen at random to destinations in 100 random prefixes each, accuracy of both path-prediction techniques and Vivaldi in predicting 10 closest destinations (in terms of delay).

PlanetLab nodes chosen at random to destinations in 100 random prefixes each. My ability to rank paths using either path prediction technique is significantly better than that of Vivaldi.

Detecting better latency detours is one of the several applications enabled by my prediction model. To evaluate how well my latency estimates preserve detours, I consider all measured paths from 35 randomly chosen out of 158 PlanetLab nodes and compute the benefits that can be obtained for these paths by taking a one-hop detour through any of the remaining 123 nodes. I estimate the latencies for every path from the chosen 35 nodes assuming the atlas consists of traces gathered from all other PlanetLab nodes plus paths to 10 random prefixes from this node. In this experiment I use the path composition approach to make predictions. I then determine for each path the best latency detour path via the remaining 123 PlanetLab nodes.



(a)



(b)

Figure 6.13: (a) Benefits and (b) Losses of choosing detours using predicted latencies. Both graphs evaluate detours from 35 PlanetLab nodes to arbitrary destinations via 123 other nodes. Note that both graphs have a log-scale y axis.

Figure 6.13 compares the predicted detour benefits against the true benefits that exist. First, the actual detour benefits on the paths I chose for evaluation are comparable with the benefits cited in [47]. Second, my latency estimates closely model the ground truth. For significant ratios of detour benefit, the fraction of paths that I predict to have such a benefit closely matches the actual number. Though I predict a detour that does not exist for 10% of the paths, only for 1% of all paths the chosen detour stretches the latency of the path by more than 30%.

Further, I evaluated how predictive of path performance are my estimates of latency and loss rate in combination. The desired property of these estimates is that they help distinguish between paths with good and bad performance. I compared the order of all measured paths from each PlanetLab node in terms of measured and predicted performance. For each node, I ranked all other nodes in terms of TCP throughput, considering throughput to be inversely proportional to latency and the square root of loss, using the standard formula for TCP throughput [61]. These rankings were computed independently using measured path properties and using my predictions for these properties. Figure 6.14 plots the correlation coefficient between the actual and predicted rankings across all PlanetLab nodes. For 80% of the nodes, the correlation coefficient is greater than 0.7. I consider rank correlation coefficient rather than the simple correlation coefficient for this experiment because the value of iPlane to distributed applications is more in ordering paths in terms of performance than producing precise estimates of path properties.

6.6 Summary

Once the forward and reverse paths between a pair of end-hosts has been predicted, one can estimate the performance between these hosts by composing the properties of the links on these paths. However, measuring the properties of all links in the Internet involves two challenges—path asymmetry and scalability.

In this chapter, I proposed and implemented techniques for scalably measuring the latency and loss rates of all inter-cluster links in the measured Internet atlas. My evaluation showed that my latency measurement techniques that account for path asymmetry yield significantly better link latency estimates than those obtained with the straightforward ap-

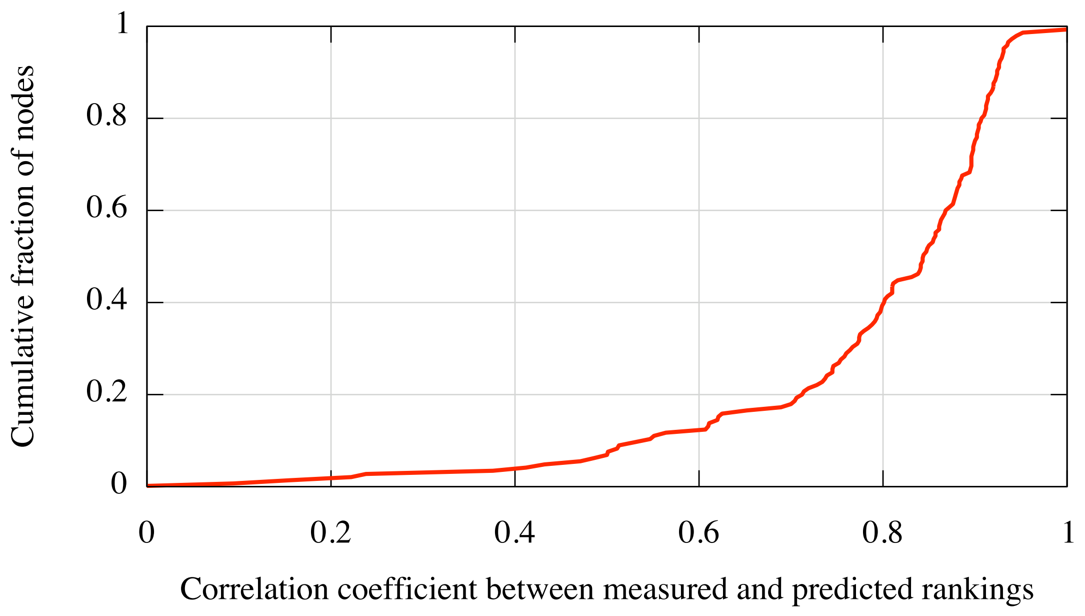


Figure 6.14: Rank correlation coefficient between measured and predicted TCP throughput for paths between PlanetLab nodes.

proach of subtracting RTTs from traceroutes. Also, using the measured link latencies and loss rates in combination with the path prediction techniques outlined in previous chapters, I obtained reasonably accurate estimates of path properties, outperforming popular coordinate-based systems such as Vivaldi. My estimates for latency and loss rate were seen to be particularly effective at ranking paths from a source to various destinations and at identifying the latency benefits that detour routing can provide.

All of the evaluation results in this chapter are however limited to the validation datasets I consider. Almost all my validation of latency and loss rate estimates are from PlanetLab nodes. The number of PlanetLab sites—around 300—is orders of magnitudes smaller than the roughly 100K prefixes at the Internet's edge, and most PlanetLab nodes are hosted at well-provisioned educational institutions. Therefore, since the validation set is not necessarily representative, my results cannot necessarily be generalized. Investigating the applicability of my results on a wider scale is future work.

Chapter 7
IMPLEMENTATION AND EVALUATION OF iPlane

In this chapter, I put together all the pieces developed in previous chapters to build and evaluate iPlane, an Internet-wide information plane. First, I provide an overview of iPlane's implementation and evaluate its practicality. Second, I describe the interfaces exported by iPlaneand how applications can use these interfaces to get information about path properties on the Internet. Finally, I use three example distributed applications—content distribution, peer-to-peer filesharing, and voice-over-IP—to demonstrate that information from iPlane can help improve application performance.

7.1 iPlane: An Information Plane

iPlane is a prototype Internet-wide information plane that I have been running and maintaining since June 2006. iPlane uses BGP paths from RouteViews [53], RIPE [28], GEANT [36], and Abilene [90] to determine the set of routable prefixes in the Internet. Once every six months, iPlane refreshes its list of prefixes and probes the .1 address in every /24 address range covered by these prefixes to discover targets for its measurements. Only addresses responsive to probes are considered targets. Currently, of the 275K prefixes observed in iPlane's database of BGP paths, iPlane has targets in 142K prefixes.

iPlane daily performs traceroutes from every PlanetLab site to a .1 in a random /24 in each of 142K prefixes. It also performs traceroutes to a random subset of these targets from each of around 500 public traceroute servers. All of these traceroutes together constitute the Internet atlas for iPlane's path predictions both in the path composition and graph-based models of prediction. The measurements necessary to cluster the interfaces observed into PoPs are performed once every few months. A typical day's traceroutes span around 67K clusters and 400K inter-cluster links.

iPlane also performs daily measurements of the loss rates of the inter-cluster links observed on the traceroutes issued that day. It performs frontier search on the gathered atlas, and it distributes the measurement tasks across the PlanetLab nodes. In the current deployment, iPlane's centralized agent schedules and monitors roughly 2700K loss rate measurements per day, a management load that a single centralized agent can easily bear. A similar setup is used to collect measurements of link latencies, but those measurements are performed only once every few months because unlike loss rate, link latency does not depend on the workload borne by the link. Fault tolerance of the central node is an issue in distributing and coordinating these measurements across PlanetLab nodes. This can be addressed by a simple failover mechanism to a standby controller.

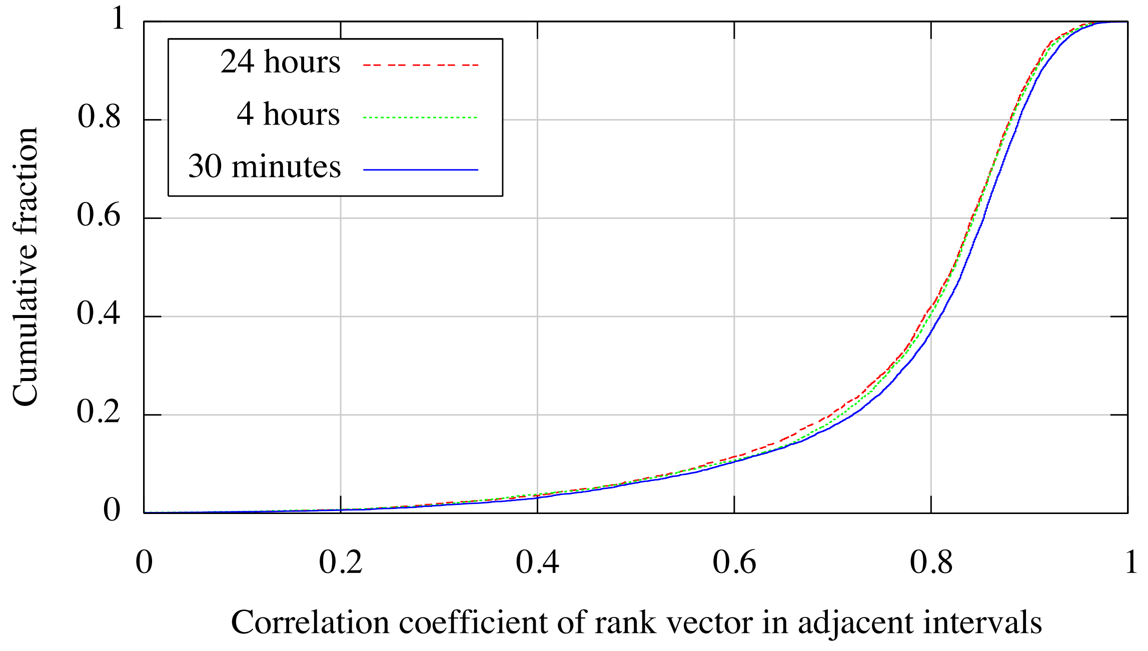
7.2 Evaluation

I next evaluate iPlane from two perspectives. First, I study the stationarity of path properties over time to evaluate if it suffices for iPlane to refresh its map of the Internet once a day. Second, since iPlane can be feasible only if the measurement load it imposes on its vantage points is reasonable, I evaluate the rate at which iPlane sources measurement traffic.

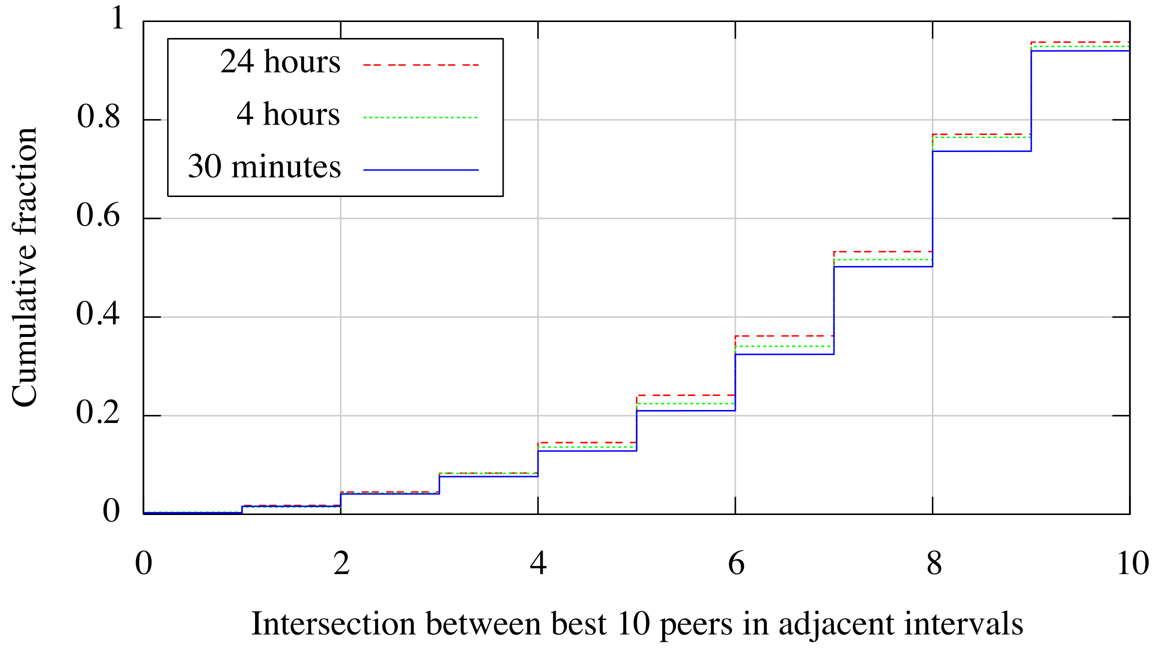
7.2.1 Stationarity of Measurements

iPlane's measurements change over time with changes in the routes in the Internet and the traffic they carry. I use PlanetLab data to estimate whether it suffices for iPlane to update its map once a day. Evaluating the stationarity of path properties for non-PlanetLab destinations is future work.

Over a period of 2 days, I measured the latency and loss rate between PlanetLab nodes once every 30 minutes. For this study, I used a dataset of 174 PlanetLab sites spanning 29 countries. In every interval, I computed for each node the ranking of all other nodes in terms of TCP throughput, considering throughput to be inversely proportional to latency and square root of loss rate. To evaluate the flux in path properties over a 30 minute timescale, I compared these rankings between adjacent 30 minute intervals. For each PlanetLab node, I computed the correlation coefficient between the ranking vectors from adjacent intervals as well as computed the intersection between the top 10 nodes in these ranking vectors. To compare this with the flux in measurements over longer timescales, I also performed these



(a)



(b)

Figure 7.1: Stationarity of measurements over different intervals over the course of a day.

computations across intervals 1 hour, 2 hours, 4 hours, 8 hours, 16 hours and 24 hours apart.

Figure 7.1(a) shows that the median correlation coefficient between the rankings is greater than 0.8 across all intervals from 30 minutes to a day. Similarly, Figure 7.1(b) shows that in the median case, 7 of the top 10 nodes in this ranking are identical on timescales from 30 minutes to a day. Though these results are only for paths between PlanetLab nodes, they seem to indicate that there is little value in updating the map more frequently than once a day, compared to once every 30 minutes. I evaluate stationarity in terms of ranking of paths rather than stationarity in terms of actual measured or predicted values because the information from iPlane is primarily intended to help distributed applications differentiate between good and bad paths.

7.2.2 Scalability

I now discuss the measurement load required for iPlane to generate and maintain a frequently refreshed map of the Internet. Even though the current deployment refreshes path and link properties once a day, with the clustering and latency measurements executed even less often, I quantify the measurement load that iPlane would impose if the refresh frequency of link properties, such as loss rate, is increased to once every 6 hours.

The measurement tasks performed by iPlane have two primary objectives—mapping of the Internet's cluster-level topology and determination of the properties of each link in the measured topology. Measurement of link properties incurs higher measurement overhead when compared to the probe traffic needed to perform a traceroute, but scales better. With more vantage points, the topology discovery traffic per node remains the same, but the overhead per node for measuring link metrics scales down, allowing the same fidelity for less overhead per node. This is even though the total number of links discovered would scale up with increase in vantage points. The measurement load associated with each technique in iPlane's measurement apparatus is summarized in Table 7.1, assuming the availability of 400 PlanetLab nodes at 200 sites. These numbers show that iPlane can produce an updated map of the Internet's routing topology every day with as little as 10Kbps of probe traffic per vantage point, and update the map of link-level attributes once every 6 hours

Table 7.1: Complexity of measurement techniques used in iPlane, based on the following assumptions. A UDP/ICMP probe is 40 bytes. A traceroute incurs a total of 500B on average. The per-link loss rate measurements require 100KB of probe traffic. Topology mapping measurements are performed from one node in each of the 200 PlanetLab sites typically up, and loss rate measurements are performed from all the PlanetLab nodes (roughly 400) at these sites. One node each at 100 PlanetLab sites suffice for clustering measurements.

Measurement Task	Tool / Technique	Frequency	Probing rate / node
Topology Mapping	*traceroute*	Once a day	200 vantage points $\times$ 150K prefixes $\times$ 500B/measurement — 7Kbps
Clustering	UDP probes for source-address-based alias resolution, ICMP-ECHO probes for reverse TTLs	One day every week	100 vantage points $\times$ 800K interfaces $\times$ 80B/measurement — 6Kbps
Latency measurements	"frontier" algorithm applied to cluster-level topology for path assignment, UDP, IP timestamp, and record route probes for symmetry, spoofed probes for one-way delays	One day every week	400 vantage points $\times$ 4000 links $\times$ 160B/measurement — 0.06Kbps
Loss rate measurements	"frontier" algorithm for path assignment, TTL-limited probes for loss rate	Continuous (every 6 hours)	400 vantage points $\times$ 4000 links $\times$ 100KB/measurement — 150Kbps

with around 150Kbps of probe traffic per vantage point. This shows that iPlane imposes minimal overhead on the vantage points in its current deployment and suggests that iPlane can refresh the Internet map more frequently.

Note the measurement overhead numbers presented above are without any optimizations. For example, properties of largely static portions of the Internet need to be probed less often. However, I currently do not attempt to identify and implement such potential optimizations.

7.3 Query Interface

The query interface exported by iPlane must be carefully designed to enable a diverse range of applications. My current implementation of the query interface exposes a database-like view of path properties between every pair of end-hosts in the Internet. For every source-destination pair, the interface returns a row with iPlane's predicted path between the hosts and the predicted latency and loss rate along this path. Any query to iPlane involves an SQL-like query on this view – selecting some rows and columns, joining the view with itself, sorting rows based on values in certain columns, and so on. The database view is merely an abstraction. iPlane does not compute a priori the entire table comprising predictions for every source-destination pair; instead it derives necessary table entries on-demand.

For example, a content distribution network (CDN) can determine the closest replica to a given client by selecting those rows that predict the performance between the client and any of the CDN's replicas. A suitable replica can then be determined by sorting these rows based on a desired performance metric. To choose a good detour node for two end-hosts to conduct VoIP, the rows predicting path properties from the given source can be joined with the set of rows predicting path properties for the given destination. A good detour is one that occurs as the destination in the first view and as the source in the second view, such that the composed performance metrics from these rows is the best. These queries can be invoked in either of the following two ways.

Query an iPlane Server: Applications that do not wish to incur the costs of downloading the annotated map and keeping it up-to-date, can query a remote iPlane service node. To support such applications, iPlane replicates the annotated map of the Internet across a bunch of machines and then provides an RPC interface to the data. Further, as some applications might need to make multiple back-to-back queries to process iPlane's measurements, we assist the application in lowering its overheads by allowing it to upload a script that can make multiple local invocations of iPlane's library. The current implementation requires that this script be written in Ruby, as Ruby scripts can be executed in a sandboxed environment and with bounded resources [83]. The output of the script's execution is returned as the response to the RPC.

The iPlane query server implements the path composition technique for predicting routes and path properties. I evaluated the overhead of the query server by issuing queries for 1000 source-destination pairs. I issued the queries via RPC from the machine hosting the server so as to rule out overhead introduced by network latency. When the query server uses the raw atlas of traceroutes, it has to read portions of the atlas from disk on-demand because the size of the raw atlas is of the order of gigabytes and cannot be loaded into memory. On a commodity machine with 4GB of memory, when using an atlas of paths from all PlanetLab nodes to all BGP prefixes, the average time taken to process a query for a source-destination pair is of the order of a second. A more complete atlas will only further increase the query overhead. However, the CLUSTER-TREE algorithm enables the query server to load the atlas into memory and, as a result, reduces the average overhead per query to the order of a millisecond.

Download the Internet Map and Process Locally: I have also implemented a library that provides an interface to download the current snapshot of the entire annotated Internet map, to process the annotated map, and to export the above SQL-like view. An application simply links against and invokes the library locally.

The library implements the graph-based prediction technique so as to minimize the size of the Internet map that a client needs to download. I repeated the same experiment as above to evaluate the latency overhead of the library. In this setting, the average time taken to process the query for a source-destination pair is of the order of 500ms. The high overhead compared to the query server, even though the memory footprint of the graph-based atlas is significantly smaller, is due to the Dijkstra-style path prediction algorithm. The graph-based prediction algorithm traverses most of the nodes in the atlas in predicting the path between any pair of end-hosts, whereas the path composition technique only considers nodes on paths to and from the source and destination. Reducing the query overhead of the library is future work. However, even the current overhead of several hundred milliseconds might suffice since the library is intended for the setting where each end-host processes its queries locally, whereas the query server receives remote queries.

7.4 Application Case Studies

My evaluation in Section 6.5 showed that my estimates of path properties such as latency and loss rate are reasonably accurate. However, since the proof is in the pudding, the real test for iPlane's estimates is in whether it improves the performance of distributed applications. Rather than being perfectly accurate in estimating individual path metrics, iPlane's accuracy needs to be good enough to help applications. Arguably, since iPlane's goal is to help any end-host in a distributed application differentiate good paths from bad paths, it is crucial for iPlane to get right its predictions from the end-host to nearby destinations rather than to precisely predict metrics to destinations far away.

In this section, I show that several popular applications can benefit from using iPlane. I evaluate three representative distributed services for potential performance benefits from using iPlane—peer-to-peer filesharing, content distribution, and voice-over-IP.

7.4.1 BitTorrent

I next show how iPlane can enable informed peer selection in popular swarming systems like BitTorrent. In current implementations, a centralized BitTorrent *tracker* serves each client a random list of peers. Each client enforces a tit-for-tat bandwidth reciprocity mechanism that incents users to contribute more upload bandwidth to obtain faster downloads. The same mechanism also serves to optimize path selection at a local level—peers upload to many random peers and eventually settle on a set that maximizes their download rate. Because reasoning about peer quality occurs locally at each client, each client needs to keep a large pool of directly connected peers (60–100 for typical swarms) even though at any time only a few of these (10–20) are actively engaged in data transfer with the client. This overhead and

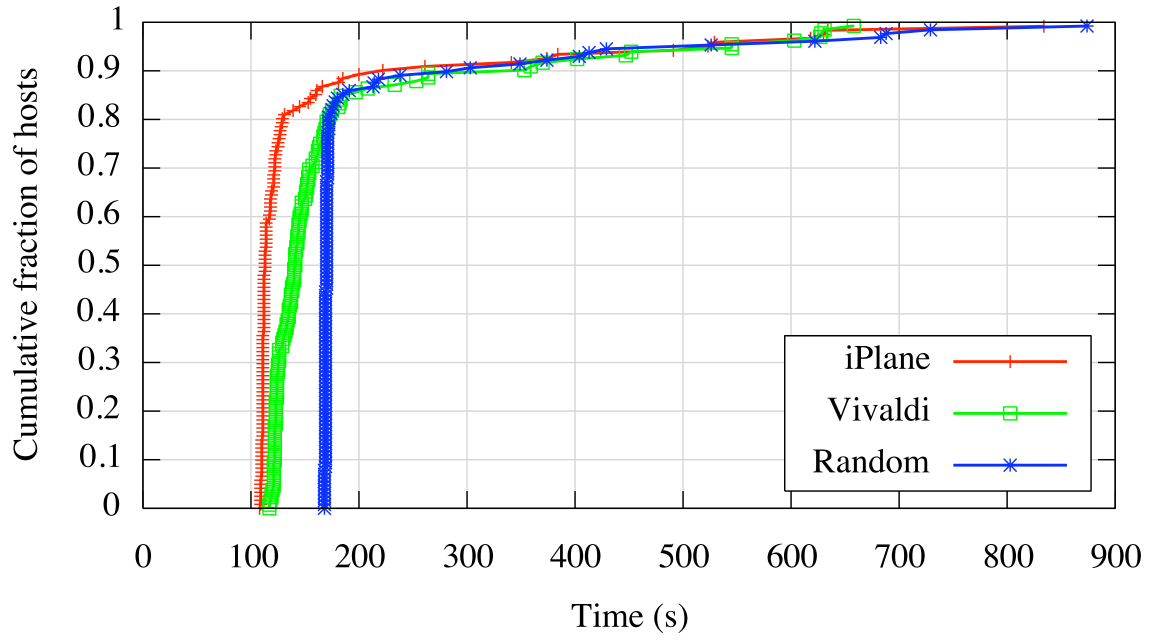


Figure 7.2: Comparison of BitTorrent download completion times with and without informed peer selection at the tracker. Informed peer selection is performed both using iPlane's estimates of latency and loss rate, and using Vivaldi's estimates of latency.

consequent delayed convergence is fundamental: with only local information, peers cannot reason about the value of neighbors without actively exchanging data with them. iPlane's predictions can overcome the lack of prior information regarding peer performance and can thus enable a clean separation of the path selection policy from the incentive mechanism.

I built a modified tracker that uses iPlane for informed peer selection. Instead of returning random peers, the tracker uses iPlane's loss rate and latency estimates to infer TCP throughput. It then returns a set of peers, half of which have high predicted throughput and the rest randomly selected. The random subset is included to prevent the overlay from becoming disconnected (e.g., no US node preferring a peer in Asia).

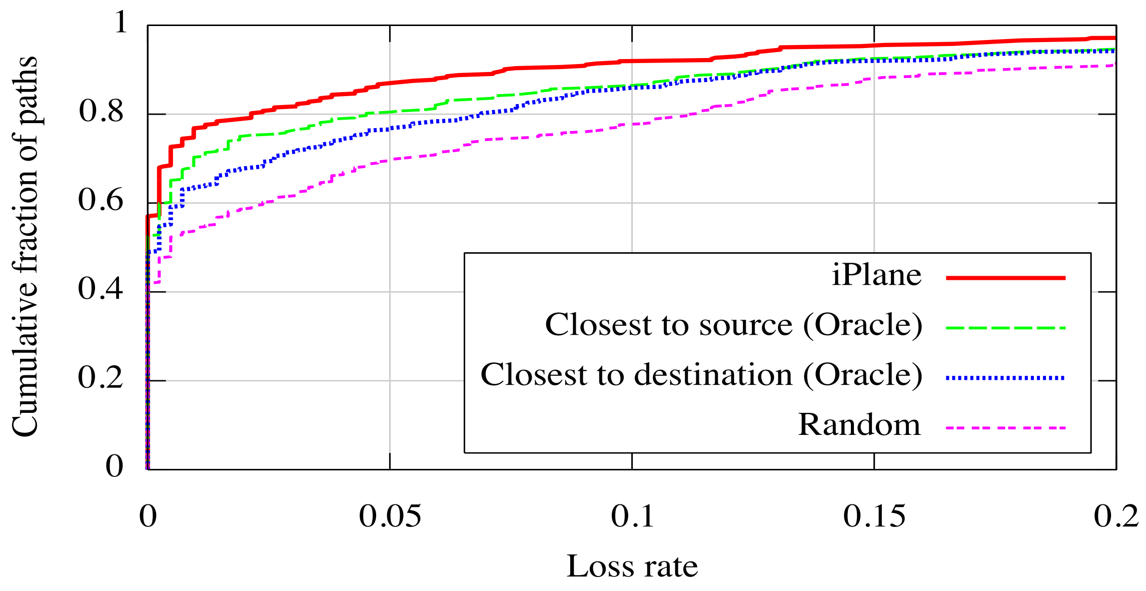
I used my modified tracker to coordinate the distribution of a 50 megabyte file over 150 PlanetLab nodes. I measured the time taken by each of the peers to download the file after the seed was started, with iPlane predictions against those of peerings induced by Vivaldi coordinates [18] and an unmodified tracker. Figure 7.2 is the result. Informed peer selection causes roughly 85% of the peers to have lower download times than default BitTorrent and around half the peers do significantly better when using latency and loss rate from iPlane than just latency from Vivaldi.

These performance improvements in BitTorrent could however be specific to the particular setting of my experiment. In real swarms, peers arrive and leave independently. Also, unlike PlanetLab nodes, end-hosts participating in BitTorrent typically are behind broadband access links, which have significantly lower bandwidth than on PlanetLab and have access links with highly variable queueing delays. I would need to explicitly measure and incorporate properties of access links, e.g., using BitProbes [37], into iPlane's prediction model to perform peer selection in BitTorrent in such settings.

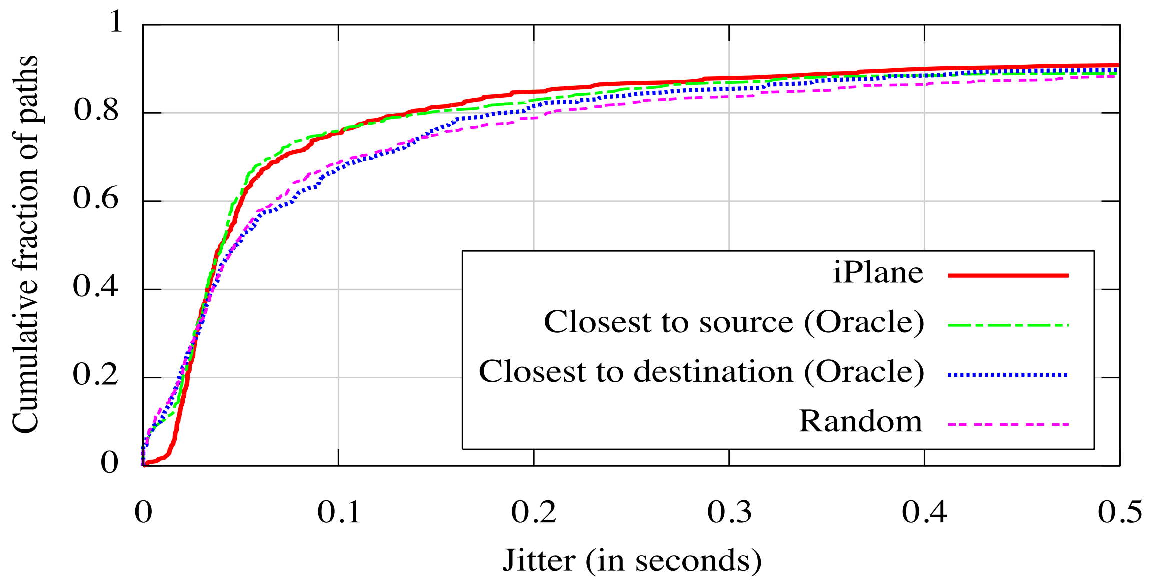
7.4.2 Voice-over-IP

Voice over IP (VoIP) is a rapidly growing application that benefits from using paths with low latency, loss, and jitter. Several VoIP implementations such as Skype [77] use relay nodes to connect end-hosts behind NATs/firewalls. Choosing the right relay node is crucial to providing acceptable user-perceived performance [68]. Reducing end-to-end latency is important since humans are sensitive to delays above a threshold. Low loss rates improve sound quality and reduce throughput consumed by compensating codecs. Measures of user-perceived sound quality such as *mean opinion score* [87] have been shown to be highly correlated with loss rate and end-to-end delay. Thus, VoIP applications can benefit from iPlane's predictions of latency and loss rate in choosing the best possible relay node.

To evaluate iPlane's ability to successfully pick good relay nodes, I emulated VoIP traffic patterns on PlanetLab. I considered 384 pairs of PlanetLab nodes, chosen at random, as being representative of end-hosts participating in a VoIP call. Between each pair, I emulated a call by sending a 10KBps UDP packet stream via another PlanetLab node chosen as the relay node. I tried 4 different relay options for each pair chosen based on (i) iPlane's estimates of latency and loss rate, (ii) latency to the source, (iii) latency to the destination, and (iv) random choice. The iPlane-informed choice was obtained by first querying for the



(a)



(b)

Figure 7.3: Comparison of (a) loss rate and (b) jitter with and without use of iPlane for end-to-end VoIP paths.

10 relay options that minimize end-to-end loss and then choosing the one that minimized end-to-end delay among these options.

Each emulated call lasted for 60 seconds, and the end-to-end loss rate and latency were measured. Figure 7.3(a) shows that significantly lower loss rates were observed along relay paths chosen based on iPlane's predictions. Additionally, Figure 7.3(b) shows that iPlane also helps to reduce jitter, which I computed as the standard deviation of end-to-end latency. These results demonstrate the potential for the use of iPlane in VoIP applications.

7.4.3 Content Distribution Network

Content distribution networks (CDNs) such as Akamai, CoDeeN, and Coral [1, 94, 25] redirect clients to a nearby replica. The underlying assumption is that distance determines network performance. However, there is more to network performance than just distance, or round trip time. TCP throughput, for example, depends on both distance and loss rate [61, 10]. Even for small web documents, loss of a SYN or a packet during slow start can markedly inflate transfer time. A CDN using iPlane can track the RTT and loss rate from each replica to the rest of the Internet. The CDN can then arrange for its name servers to redirect the client to optimize using the model of its choice.

I emulate a small CDN comprising 30 randomly chosen PlanetLab nodes. Each node serves 3 files of sizes 10KB, 100KB, and 1MB. I use 141 other PlanetLab nodes to emulate clients. Each client downloads all 3 files from the replica that provides the best TCP throughput as predicted by the PFTK model [61] using iPlane's estimates of RTT and loss rate, and from the replica closest in terms of *actual* measured RTT. Figure 7.4 compares the download times experienced by the clients, excluding the latency of redirecting to the replica. Choosing the replica for optimized TCP throughput based on iPlane's predictions provides slightly better performance than choosing the closest replica.

7.5 Summary

I put together the path prediction and link measurement techniques from previous chapters to build iPlane, an Internet-wide information plane. iPlane builds an Internet atlas by performing traceroutes daily from several hundred geographically distributed vantage

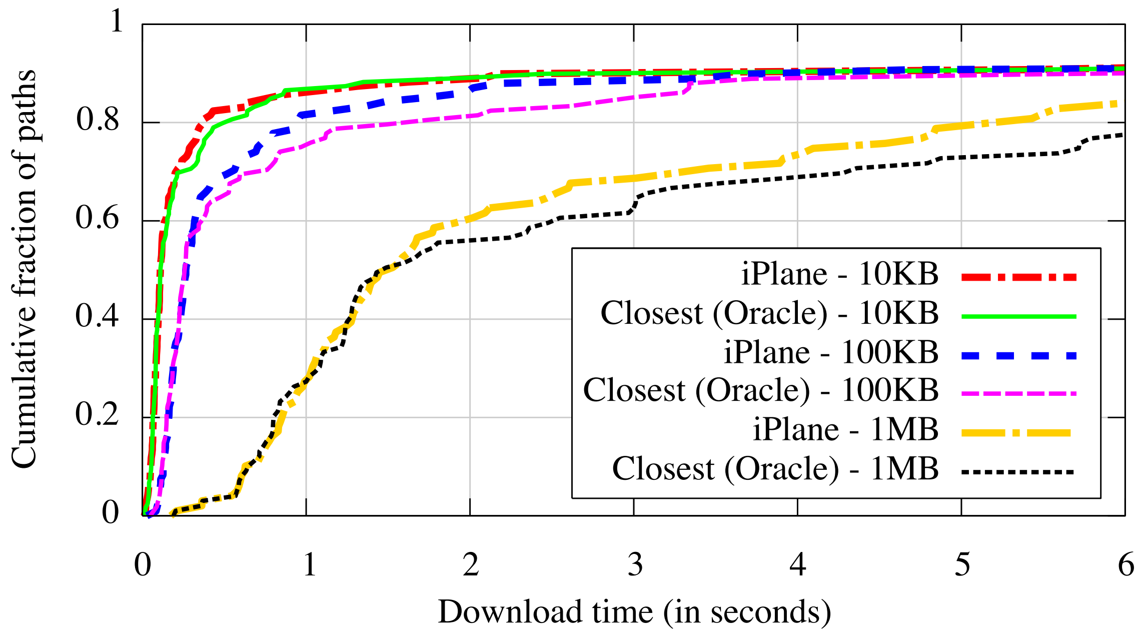


Figure 7.4: Comparison of download times from replicas in the CDN chosen by iPlane and from replicas closest in terms of latency. Each download time is the median of 5 measurements.

points—PlanetLab nodes and public traceroute servers—to destinations in several hundred thousand prefixes. It hosts a query server that implements the path composition approach to provide predictions of routes and path properties between arbitrary end-hosts. iPlane also implements a library that enables end-hosts to download the graph-based atlas and process queries locally.

My evaluation of iPlane demonstrated that use of iPlane's estimates for path selection in distributed applications can improve performance. In a BitTorrent swarm for a 50MB file over 150 PlanetLab nodes, 85% of peers had faster download times than default BitTorrent when using iPlane's latency and loss rate estimates to choose peers. In voice-over-IP, iPlane helped choose detour paths with significantly lesser latency and jitter than choosing at random. In a content distribution network, use of iPlane's estimates to choose replicas performed as well as choosing replicas assuming *a priori* knowledge of latencies, and in many cases even better.

Chapter 8
CONCLUSIONS AND FUTURE WORK

In this chapter, I summarize the work presented in this dissertation, review the thesis it supports and the contributions made, and outline some lines of future work that would build upon this dissertation.

8.1 Thesis and Contributions

In this dissertation, I supported the following thesis: *it is practical to build a scalable system that measures the Internet from end-hosts and synthesizes these using a topology-aware methodology to estimate the latency and loss rate between arbitrary end-hosts on the Internet, with accuracy sufficient to improve the performance of distributed applications.*

The work presented in this dissertation made the following contributions:

Algorithms for predicting the route through the Internet between arbitrary end-hosts. I develop algorithms that predict the route through the Internet between any arbitrary pair of end-hosts given a few measurements of the Internet's routing topology from the source and/or from the destination. I present and evaluate algorithms for doing so in two scenarios—first, when the input is a set of Internet routes, and second, when the input is a set of links that capture the Internet's structure. The two scenarios present a trade-off in accuracy versus the size of input. My algorithms for predicting routes do so at the right granularity required by the information plane—at the granularity that helps capture Internet performance. I do so by clustering together portions of the Internet are similar from a routing perspective.

Scalable techniques for measuring the properties of Internet links. I develop measurement techniques for estimating the latencies of links in the Internet. Like existing Internet measurement tools, these techniques exploit properties of protocols in the Internet in novel ways. I also construct a simple algorithm that distributes the measurement load across end-hosts from which the information plane can issue measurements.

Design and implementation of iPlane, an Internet-wide information plane. I construct an information plane, iPlane, that is usable today by applications. iPlane constantly issues measurements from over 200 PlanetLab nodes and over 500 public traceroute servers to maintain a daily updated snapshot of the Internet's structure annotated with link metrics. iPlane exports an interface that end-hosts can use to contribute measurements, and an interface that applications can use to query for information about path properties.

Demonstration of iPlane's utility to distributed applications. My evaluation of the utility of information provided by iPlane uses three representative applications—content distribution, peer-to-peer filesharing, and voice-over-IP. For each of these applications, I demonstrate that application performance significantly improves when information about the network is available from iPlane.

8.2 Key Ideas

The following key ideas enable me to build iPlane and support the stated hypothesis.

- **Structural approach:** Prior attempts at estimating path properties typically treat the Internet as a blackbox, relying on end-to-end measurements as input. Instead, I discover and utilize the Internet's structure for making inferences. To estimate the path properties between any pair of end-hosts, I predict the route between them and then compose the properties of links and path segments along the predicted route to estimate end-to-end path properties.

- **Path composition:** The primary technique I employ to predict the route between a source and a destination is to compose a path segment observed on a route from the source with a path segment seen on a route to the destination. Path composition is based on the principle of route similarity—since routing in the Internet is predominantly destination-based, two nearby sources are likely to have similar routes to the same destination.

- **Path selection:** The rich connectivity of the Internet's structure, however, results in the path composition technique yielding several candidate predicted routes from a source to a destination. To distinguish amongst these candidate routes, I infer and apply routing policy used by ISPs. I infer routing policy for each ISP in two forms: 1) its preferences amongst its neighbors, and 2) the pairs of neighbors between which it provides transit.

- **Tackling path asymmetry:** The structural approach necessitates the inference of properties of links and path segments. This task is complicated by the fact that iPlane is restricted to issuing measurements from a limited set of vantage points and hence can only gather measurements of the round-trip path to end-hosts and routers that it does not control. Since routes in the Internet can be asymmetric, it is tough to estimate properties of a link or a path segment observed only on the forward path based on round-trip measurements. In this dissertation, I develop techniques to identify when routes are symmetric and use these techniques to measure link latencies.

- **Clustering:** Measuring the network at Internet-scale has been largely believed to be infeasible. In building iPlane, I demonstrate that path properties between any arbitrary pair of end-hosts can be updated once every few hours by operating at the right granularity. Though the Internet has millions of end-hosts and routers, these can be clustered into roughly 50K BGP atoms [7] and 100K Points-of-Presence. By clustering elements of the Internet that are similar from a routing and performance perspective, I manage to obtain Internet-wide coverage in building an information plane.

8.3 Future Work

The work presented in this dissertation can be extended in several ways.

8.3.1 Improving iPlane's Accuracy

My prototype implementation of iPlane has demonstrated that it is feasible to make reasonably accurate estimates of routes and path properties between arbitrary end-hosts. Measuring at Internet-scale was previously thought to be untenable. My work demonstrated that measuring at the right granularity—at the level of PoPs within an AS—reduces the number of measurements significantly without loss of accuracy in predictions.

However, the results from my evaluation also show that there remains significant room for improvement in the accuracy of iPlane's predictions. While some of the inaccuracy could be attributed to the incompleteness of the atlas that iPlane currently uses, there are a bunch of directions in which iPlane's prediction techniques can be extended to further improve its accuracy. First, a better job could be done at processing the atlas, e.g., by improving the algorithms used to cluster router interfaces into routers and PoPs, algorithms which are currently not comprehensive. Second, the inference of routing policy from observed routes in a manner that can be represented compactly requires work beyond the AS export policies and the AS preferences that I currently infer. Third, as the number of observed paths available to iPlane increases, not only will the size of the atlas blow up but also the path composition algorithm needs to be extended to handle the fact that every vantage point will not have measurements to every prefix. Rather than relying on a complete path segment to the destination, the path prediction algorithm would need to instead stitch together multiple path segments—an algorithm in the middle of the spectrum between path composition and graph-based prediction.

8.3.2 Building Internet-scale Distributed Applications

The availability of information about the network from iPlane has the potential to greatly simplify the task of building distributed applications that scale to the Internet. iPlane provides applications access to properties of paths without each application having to implement an Internet measurement toolkit of its own.

However, experience in building applications that leverage iPlane is necessary to evaluate iPlane's ability to simplify construction of applications. First, iPlane is intended to be a single source of information across distributed applications on the Internet. But, since an application's request for information about a network resource is not atomic with its use of that resource, multiple applications can act on the same information leading to none of the applications getting the path performance that they expected. Therefore, applications can treat information obtained from iPlane only as a hint. Once an application starts using a path, it will need to monitor the performance it obtains for more fine grained information.

Second, the exercise of building applications that use iPlane will throw more light on the right API that iPlane should export. iPlane's current interface of accepting a pair of IP addresses as input and returning its prediction for the path and path properties between these addresses is the most basic interface possible. In practice, many applications will need to fetch predictions for paths between several pairs of addresses. For example, to determine the best among a set of detour nodes for communication between a source and a destination, the application will need to query for paths from the source to possible detour nodes and from all of these nodes to the destination. If such a conjugate query is common enough across applications, one could imagine iPlane supporting queries of the type *"best-detour-node"*. The right set of conjugate queries that iPlane should support will be apparent only once iPlane is applied to several distributed applications. Also, one of the reasons to build iPlane as an information plane rather than as a library was to enable it to receive feedback from applications about the information it provides. The right interface for accepting such information from applications remains an open question.

8.3.3 Scaling iPlane

iPlane currently builds an Internet atlas using measurements from a few hundred geographically distributed vantage points and refreshes this atlas once a day. iPlane's view of the Internet's structure is limited by its set of vantage points, and it may miss several changes in the Internet owing to the frequency with which it updates its information. iPlane needs to scale both in space as well as time.

To scale iPlane in terms of its network presence, it is desirable to have one vantage point in every edge prefix. To do so, end-hosts will need to be recruited to participate as measurement vantage points. Previous Internet measurements efforts such as DIMES [75] and NETI@home [12] have attempted to recruit end-hosts as agents by appealing to the scientific spirit of users, which has helped them assemble a few thousand vantage points. Expanding such measurement platforms further instead requires that users be provided with some incentive for downloading and installing Internet measurement software on their computers.

One way to incent users to contribute Internet measurements would be to couple a measurement agent with an application that users desire. As seen in my evaluation, end-hosts can significantly improve iPlane's predictions for paths from that end-host by contributing a few measurements. Thus, if users adopt an application that uses iPlane, their incentives would be aligned to contribute measurements and as a result increase their own application performance. Further, since every end-host needs to contribute extremely few measurements, the measurement load imposed on any given end-host would be orders of magnitude less than that on iPlane's dedicated vantage points, e.g., PlanetLab nodes.

While one approach to keep iPlane's view of the Internet up-to-date would be to have all of the vantage points repeat the measurements assigned to them more frequently, such an approach intrinsically adds overhead. Instead, keeping the atlas up-to-date should be driven by the observation that not all parts of the Internet are constantly in flux. At any point in time, only the properties of some portions of the Internet change and only these changes need to be measured to keep the atlas up to date. Therefore, iPlane's current mode of repeating all of its measurements regularly must be replaced by a two-tier system—lightweight mechanisms to detect change (or the rate of change) followed by the triggering of more comprehensive measurements to incorporate the change in the atlas. For example, to update its map of the Internet's topology, iPlane currently issues traceroutes to one target in every prefix daily. Instead, prefixes to which routes have changed can be detected using passive monitoring of BGP feeds or by tracking the return TTLs on pings, and traceroutes can be issued only to such prefixes. Similarly, paths that are unchanged on a daily basis can be probed less often and those that change more often, self-tuning the monitoring rate to the system flux.

8.3.4 Longitudinal Analysis of the Internet

Apart from providing real-time information about the network to applications, the data gathered by iPlane over time is a great dataset for longitudinal analysis of the Internet. iPlane has been refreshing its Internet atlas once a day for more than two years now, and this dataset continues to grow. While there have previously been efforts to continually gather and maintain measurements of the Internet, e.g., RouteViews [53], Skitter [9], the data gathered by iPlane provides significantly more coverage than prior efforts.

iPlane's data has already been tapped for several studies, e.g., to study the evolution of the Internet's AS-level topology [58]. However, there remain a large number of features that can be mined from this data. For example, the topology data gathered by iPlane can be used to study the stability of routes on the Internet. The last such study performed was by Paxson [63] in 1997 using traceroutes gathered between all pairs of nodes in the 30-odd node NIMI testbed. Not only has the Internet evolved drastically over the last decade, but iPlane's atlas of routes—from several hundred vantage points to several hundred thousand prefixes—is much more comprehensive than the data used by Paxson. Studying the stability of Internet routes is also necessary to develop the right strategy that iPlane must employ to maintain an up-to-date atlas.

8.4 Summary

My work has leveraged the expertise built up by the Internet measurement community to make it easier to build distributed applications. iPlane provides information about paths and path properties between arbitrary end-hosts on the Internet, and thus, eliminates every application from having to reimplement a network measurement toolkit. While my current implementation of iPlane provides reasonably accurate predictions of path properties, much remains to be done to improve iPlane's prediction accuracy and its coverage. I hope my work in building iPlane will herald a new model for building applications—much as applications assume the easy availability of packet delivery today, iPlane enables applications to assume the easy availability of information about the network. This new model has the potential to spawn a new generation of Internet-scale distributed applications.

BIBLIOGRAPHY

[1] Akamai, Inc. home page. `http://www.akamai.com`.

[2] Aditya Akella, Bruce Maggs, Srinivasan Seshan, Anees Shaikh, and Ramesh Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 353–364, Karlsruhe, Germany, August 2003.

[3] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Banff, Alberta, Canada, October 2001.

[4] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 153–158, Rio de Janeiro, Brazil, October 2006.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. IETF RFC 2475, October 1998.

[6] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: An overview. IETF RFC 1633, June 1994.

[7] Andre Broido and kc claffy. Analysis of RouteViews BGP data: policy atoms. In *Proceedings of the ACM SIGMOD/PODS Workshop on Network-Related Data Management*, Santa Barbara, CA, May 2001.

[8] Ramón Cáceres, N. G. Duffield, Joseph Horowitz, and Donald F. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, 45(7):2462–2480, November 1999.

[9] CAIDA. Skitter project. `http://www.caida.org/tools/measurement/skitter/`.

[10] Neal Cardwell, Stefan Savage, and Thomas Anderson. Modeling TCP latency. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1742–1751, Tel Aviv, Israel, March 2000.

[11] Martin Casado, Tal Garfinkel, Weidong Cui, Vern Paxson, and Stefan Savage. Opportunistic measurement: Extracting insight from spurious traffic. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, College Park, MD, November 2005.

[12] Jr. Charles Robert Simpson and George F. Riley. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, pages 168–174, Antibes Juan-les-Pins, France, April 2004.

[13] Yan Chen, David Bindel, Hanhee Song, and Randy H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 55–66, Portland, OR, September 2004.

[14] Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, 2002.

[15] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. A knowledge plane for the Internet. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 3–10, Karlsruhe, Germany, August 2003.

[16] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, 2003.

[17] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet coordinates for distance estimation. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 178–187, Tokyo, Japan, March 2004.

[18] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 15–26, Portland, OR, September 2004.

[19] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, and George Riley. AS relationships: inference and validation. *ACM Computer Communication Review*, 37(1):29–40, 2007.

[20] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 43–56, San Diego, CA, October 2007.

[21] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. Efficient algorithms for large-scale topology discovery. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 327–338, Banff, Alberta, Canada, June 2005.

[22] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 915–923, Anchorage, AK, April 2001.

[23] eMule project. `http://www.emule-project.net`.

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

[25] Michael J. Freedman, Eric Freudenthal, and David Mazières. Democratizing content publication with Coral. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 239–252, San Francisco, CA, March 2004.

[26] Michael J. Freedman, Karthik Lakshminarayanan, and David Mazieres. OASIS: Anycast for any service. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 129–142, San Jose, CA, May 2006.

[27] Lixin Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, December 2001.

[28] Fotis Georgatos, Florian Gruber, Daniel Karrenberg, Mark Santcroos, Ana Susanj, Henk Uijterwaal, and René Wilhelm. Providing active measurements as a regular service for ISPs. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, 2001.

[29] Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. IrisNet: An architecture for a world-wide sensor web. *IEEE Pervasive Computing*, 2(4):14–21, October 2003.

[30] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet map discovery. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and*

Communications Societies (INFOCOM), pages 1371–1380, Tel Aviv, Israel, March 2000.

[31] Krishna P. Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David J. Wetherall. Improving the reliability of Internet paths with one-hop source routing. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 183–198, San Francisco, CA, December 2004.

[32] Mehmet H. Gunes and Kamil Sarac. Resolving IP aliases in building traceroute-based Internet maps. *IEEE/ACM Transactions on Networking*, 2008.

[33] Ningning Hu and Peter Steenkiste. Exploiting Internet route sharing for large scale available bandwidth estimation. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 187–192, Berkeley, CA, October 2005.

[34] Ningning Hu and Peter Steenkiste. Quantifying Internet end-to-end route similarity. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, Adelaide, Australia, March 2006.

[35] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the Internet with PIER. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 321–332, Berlin, Germany, September 2003.

[36] Intel-DANTE monitoring project. `http://www.cambridge.intel-research.net/monitoring/dante/`.

[37] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Leveraging BitTorrent for end host measurements. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, pages 32–41, Louvain-la-neuve, Belgium, April 2007.

[38] Van Jacobson. Traceroute. `ftp://ftp.ee.lbl.gov/traceroute.tar.Z`.

[39] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and Jr. James W. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 197–212, San Diego, CA, October 2000.

[40] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP geolocation using delay and topology measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 71–84, Rio de Janeiro, Brazil, October 2006.

[41] Ethan Katz-Bassett, Harsha V. Madhyastha, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Studying black holes in the Internet with Hubble. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, April 2008.

[42] Kazaa. `http://www.kazaa.com`.

[43] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet routing convergence. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 175–187, Stockholm, Sweden, August 2000.

[44] Sung-Ju Lee, Puneet Sharma, Sujata Banerjee, Sujoy Basu, and Rodrigo Fonseca. Measuring bandwidth between PlanetLab nodes. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, pages 292–305, Boston, MA, March 2005.

[45] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi. Constructing Internet coordinate system based on delay measurement. *IEEE/ACM Transactions on Networking*, 13(3):513–525, June 2005.

[46] Eng Keong Lua, Timothy Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft. On the accuracy of embeddings for Internet coordinate systems. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 125–138, Berkeley, CA, October 2005.

[47] Ratul Mahajan. *Practical and Efficient Internet Routing with Competing Interests.* PhD thesis, University of Washington, 2005.

[48] Ratul Mahajan, Neil Spring, Tom Anderson, and David Wetherall. Inferring link weights using end-to-end measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 231–236, Marseille, France, November 2002.

[49] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level Internet path diagnosis. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 106–119, Bolton Landing, NY, October 2003.

[50] Ratul Mahajan, Ming Zhang, Lindsey Poole, and Vivek Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, April 2008.

[51] Yun Mao and Lawrence Saul. Modeling distances in large-scale networks by matrix factorization. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 278–287, Taormina, Sicily, Italy, October 2004.

[52] Z. Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On AS-level path inference. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 339–349, Banff, Alberta, Canada, June 2005.

[53] David Meyer. RouteViews. `http://www.routeviews.org`.

[54] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an AS-topology model that captures route diversity. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 195–206, Pisa, Italy, September 2006.

[55] Mike Muuss. Ping. `http://ftp.arl.army.mil/pub/ping.shar`.

[56] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 170–179, New York, NY, June 2002.

[57] T. S. Eugene Ng and Hui Zhang. A network positioning system for the Internet. In *Proceedings of the USENIX Annual Technical Conference*, pages 141–154, Boston, MA, June 2004.

[58] Ricardo Oliveira, Beichuan Zhang, and Lixia Zhang. Observing the evolution of Internet AS topology. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 313–324, Kyoto, Japan, August 2007.

[59] OpenView home page. `http://www.managementsoftware.hp.com/`.

[60] David Oppenheimer, Jeannie Albrecht, David Patterson, and Amin Vahdat. Scalable wide-area resource discovery. Technical Report UCB//CSD-04-1334, University of California Berkeley, July 2004.

[61] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 303–314, Vancouver, BC, Canada, September 1998.

[62] Venkata N. Padmanabhan, Lili Qiu, and Helen Wang. Server-based inference of Internet link lossiness. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 145–155, San Francisco, CA, April 2003.

[63] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[64] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 278–291, Berkeley, CA, October 2003.

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

[66] Francesco Lo Presti, N. G. Duffield, Joe Horowitz, and Don Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Transactions on Networking*, 10(6):761–775, December 2002.

[67] Jian Qiu and Lixin Gao. AS path inference by exploiting known AS paths. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, San Francisco, CA, November 2006.

[68] Shansi Ren, Lei Guo, and Xiaodong Zhang. ASAP: an AS-aware peer-relay protocol for high quality VoIP. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 70–70, Lisbon, Portugal, July 2006.

[69] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. BGP routing stability of popular destinations. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, November 2002.

[70] RIPE Routing Information Service. `http://www.ripe.net/ris/`.

[71] Sarangworld project. `http://www.sarangworld.com/TRACEROUTE/`.

[72] Stefan Savage. Sting: a TCP-based network measurement tool. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 71–79, Boulder, CO, October 1999.

[73] Stefan Savage, Thomas 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, January 1999.

[74] Scalable Sensing Service home page. `http://networking.hpl.hp.com/s-cube`.

[75] Yuval Shavitt and Eran Shir. DIMES: Let the Internet measure itself. *ACM Computer Communication Review*, 35(5):71–74, October 2005.

[76] Yuval Shavitt and Tomer Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Hong Kong, March 2004.

[77] Skype home page. `http://www.skype.com`.

[78] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Improving accuracy in end-to-end packet loss measurement. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 157–168, Philadelphia, PA, August 2005.

[79] Neil Spring, Mira Dontcheva, Maya Rodrig, and David Wetherall. How to resolve IP aliases. Technical Report 04-05-04, University of Washington Department of Computer Science and Engineering, May 2004.

[80] Neil Spring, Ratul Mahajan, and Tom Anderson. Quantifying the causes of path inflation. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 113–124, Karlsruhe, Germany, August 2003.

[81] Neil Spring, Ratul Mahajan, David Wetherall, and Tom Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[82] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using PlanetLab for network research: Myths, realities, and best practice. *ACM SIGOPS Operating Systems Review (OSR)*, 40(1):17–24, January 2006.

[83] Neil Spring, David Wetherall, and Thomas Anderson. Scriptroute: A public Internet measurement facility. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 225–238, Seattle, WA, March 2003.

[84] Neil Spring, David Wetherall, and Tom Anderson. Reverse-engineering the Internet. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, pages 3–8, Cambridge, MA, November 2003.

[85] Lakshminarayanan Subramaniam, Ion Stoica, Hari Balakrishnan, and Randy H. Katz. OverQos: An overlay based architecture for enhancing Internet QoS. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 71–84, San Francisco, CA, March 2004.

[86] Liying Tang and Mark Crovella. Virtual landmarks for the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 143–152, Miami, FL, October 2003.

[87] Shu Tao, Kuai Xu, Antonio Estepa, Teng Fei, Lixin Gao, Roch Guerin, Jim Kurose, Don Towsley, and Zhi-Li Zhang. Improving VoIP quality through path switching. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 2268–2278, Miami, FL, March 2005.

[88] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in ISP networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 313–318, Miami, FL, October 2003.

[89] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 307–319, New York, NY, June 2004.

[90] The Internet2 Observatory. `http://www.internet2.edu/observatory/`.

[91] IBM Tivoli software home page. `http://www.ibm.com/software/tivoli`.

[92] Traceroute servers. `http://www.traceroute.org`.

[93] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no longer considered harmful. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 215–230, San Francisco, CA, December 2004.

[94] Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek S. Pai, and Larry L. Peterson. Reliability and security in the CoDeeN content distribution network. In *Proceedings of the USENIX Annual Technical Conference*, pages 171–184, Boston, MA, June 2004.

[95] Mike Wawrzoniak, Larry Peterson, and Timothy Roscoe. Sophia: An information plane for networked systems. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, Cambridge, MA, November 2003.

[96] Bernard Wong, Aleksandrs Slivkins, and Emin Gun Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 85–96, Philadelphia, PA, August 2005.

[97] Bernard Wong, Ivan Stoyanov, and Emin Gun Sirer. Octant: A comprehensive framework for the geolocalization of Internet hosts. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 313–326, Cambridge, MA, April 2007.

[98] Wen Xu and Jennifer Rexford. MIRO: Multi-path interdomain routing. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 171–182, Pisa, Italy, September 2006.

[99] Xiaowei Yang, David Clark, and Arthur Berger. NIRA: A new routing architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, August 2007.

[100] Xiaowei Yang and David Wetherall. Source selectable path diversity via routing deflections. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 159–170, Pisa, Italy, September 2006.

[101] Xiaowei Yang, David Wetherall, and Tom Anderson. TVA: A DoS-limiting network architecture. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 241–252, Philadelphia, PA, August 2005.

[102] Ming Zhang, Yaoping Ruan, Vivek Pai, and Jennifer Rexford. How DNS misnaming distorts Internet topology mapping. In *Proceedings of the USENIX Annual Technical Conference*, pages 369–374, Boston, MA, June 2006.

[103] Ming Zhang, Chi Zhang, Vivek Pai, Larry Peterson, and Randy Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 167–182, San Francisco, CA, December 2004.

[104] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the characteristics and origins of Internet flow rates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 309–322, Pittsburgh, PA, August 2002.

[105] Yin Zhang, Vern Paxson, and Scott Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. Technical report, ACIRI, May 2000.

[106] Ying Zhang, Z. Morley Mao, and Ming Zhang. Effective diagnosis of routing disruptions from end systems. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, April 2008.

[107] Yao Zhao, David Bindel, and Yan Chen. Towards unbiased end-to-end network diagnosis. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 219–230, Pisa, Italy, September 2006.

VITA

Harsha V. Madhyastha was born in Bangalore, India. He received his Bachelor of Technology degree in computer science and engineering from the Indian Institute of Technology, Madras in 2003, and his Master of Science degree in computer science and engineering from the University of Washington in 2006. His research interests broadly encompass building networked systems and understanding the behavior and properties of such systems.