A Structural Approach to Latency Prediction

Harsha V. Madhyastha[*] University of Washington
Thomas Anderson University of Washington
Arvind Krishnamurthy University of Washington
Neil Spring University of Maryland
Arun Venkataramani University of Massachusetts Amherst

ABSTRACT

Several models have been recently proposed for predicting the latency of end to end Internet paths. These models treat the Internet as a black-box, ignoring its internal structure. While these models are simple, they can often fail systematically; for example, the most widely used models use metric embeddings that predict no benefit to detour routes even though half of all Internet routes can benefit from detours.

In this paper, we adopt a structural approach that predicts path latency based on measurements of the Internet's routing topology, PoP connectivity, and routing policy. We find that our approach outperforms Vivaldi, the most widely used black-box model. Furthermore, unlike metric embeddings, our approach successfully predicts 65% of detour routes in the Internet. The number of measurements used in our approach is comparable with that required by black box techniques, but using traceroutes instead of pings.

Categories and Subject Descriptors

C.2.3 [**Communication Networks**]: Architecture and Design—*topology*; C.2.5 [**Communication Networks**]: Local and Wide-Area Networks—*Internet*

General Terms

Algorithms, Design, Experimentation, Measurement

Keywords

Latency prediction, Internet topology, Route measurements

1. INTRODUCTION

The latency between Internet hosts is an important component of path and server selection in many distributed applications. Popular Internet services such as Google, Akamai, and Bittorrent use latency estimates to direct clients to nearby peers or replicas. Overlay routing systems can use latency estimates for neighbor selection in distributed hash tables, construction of efficient multicast trees, and detour routing on the Internet.

To meet this demand, several network distance estimation techniques such as IDMaps, GNP, and Vivaldi have been proposed [2, 12, 1, 17, 9]. These techniques use latency measurements from a few *vantage points* to clients to predict the latencies of paths that are not directly measured. A popular prediction methodology is to embed nodes in a low-dimensional coordinate space and use the corresponding vector distance to predict the latency between arbitrary nodes.

These techniques treat the network as an unknown "black box." Black-box approaches are desirable if they can obtain good accuracy using simple models that abstract away unimportant details of the system. However, despite their compelling simplicity, these techniques have some serious shortcomings. For example, black box techniques appear poorly suited to predicting complex path properties such as bottleneck capacity, loss and jitter. Furthermore, popular black box techniques based on metric embeddings [12, 1] are fundamentally incapable of predicting detour routes. Detours exist when underlying distances violate the triangle inequality—a common occurrence in the Internet today [15, 21].

Our position is that measuring the Internet's topology and modeling complex routing protocols is fundamental to accurately predicting its end to end performance characteristics. To this end, we develop and evaluate a *structural* model of Internet routing at the granularity of Points of Presence (PoPs); we use this model to predict latency and show that it outperforms Vivaldi, the most widely used black box model. We first measure the network to construct an "atlas" of the Internet, launching probes from vantage points distributed across the world. Clients add themselves into the atlas by contributing a small number of traceroutes, and then download predictions for the sets of IP prefixes with unique routing behavior. This is similar to how clients use a small number of measurements to position themselves today in a black box metric embedding. Once a client integrates itself to the Internet atlas, the system can make accurate predictions about the client's connectivity to other end hosts.

Our goal is to determine a *simple* structural model that predicts paths and latencies with high accuracy. To this end, we rely on the comprehensiveness of the atlas and observed routing behavior as much as possible. Our basic principle is to exploit similarity of routes. Our hypothesis is that given a large number of distributed vantage points, the route from any source to a destination will be similar to the route to the same destination from a vantage point close to the source. To abstract away details, we group geographically proximate routers using a novel clustering method. Informally, our method "zooms out" on the atlas and splices a short path segment from the source with an observed path segment going to

[*]Email: harsha@cs.washington.edu

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IMC'06, October 25–27, 2006, Rio de Janeiro, Brazil.
Copyright 2006 ACM 1-59593-561-4/06/0010 ...$5.00.

the destination as an estimate of the actual path. The latency of the path is then estimated from measured latencies of constituent segments.

A limitation of our approach is that it requires the investment of measurement effort in generating the atlas. However, the traffic overhead is small. Regenerating the entire atlas on a daily basis, a time period over which most routes are stationary [13], takes 4Kbps for each vantage point. Once the atlas is generated, new clients need make as few as ten measurements to gain its benefits.

While we focus on latency in this paper to provide a direct comparison to previous techniques, our long term goal is to use structural models to predict a broader set of path properties, such as capacity, loss rate, and available bandwidth. A subsequent paper explores this in more detail [8]. Further, we believe the trend to increased instrumentation of the Internet favors structural models. Improvements in Internet measurement will increase the coverage and quality of maps over time, and one common database can be used for different applications. A structural model is also amenable to gradual 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. These issues are left for future work.

2. STRUCTURAL MODEL

We seek to develop a simple structural model that is capable of predicting latency between an arbitrary pair of Internet hosts. As it is both infeasible and unscalable to obtain fine-grained data about the entire Internet, the model has to operate with incomplete and coarse-grained information without compromising its prediction accuracy. We seek to strike a balance between scalability and accuracy in our approach.

2.1 Overview

Our technique predicts end-to-end paths between a pair of nodes by composing *path segments*, fragments of known Internet paths. Most of these segments are obtained from an Internet atlas that is constructed and maintained by a set of distributed vantage points. The atlas comprises the topology and round trip latency measurements of the Internet core and some selected portions of the edge to the vantage points. When an end-host desires predictions, it locates itself within the atlas using a small number of measurements. These new measurements are combined with the core atlas for path prediction. To reflect asymmetry of routes, we separately predict the path in the forward and reverse directions and derive end-to-end RTT using the latency measurements of the segments traversed in each direction. These predictions can then be downloaded to the client or local server to enable fast lookups for all destinations.

The basic principle underlying our 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 co-located 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, our 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 to the destination from one of the vantage points. We hope to make accurate predictions by maximizing this overlap, subject to the way routing protocols optimize key routing metrics such as AS path length, hop counts, or latency.

The following challenges and questions are the most significant:

- What portions of the Internet should be probed by the vantage points to generate a sufficiently rich atlas? The challenge is to ensure sufficient coverage without requiring exhaustive probing.
- What is the granularity of structural information used by the atlas? If the atlas stores topology at the granularity of network interfaces, it might fail to detect the nearness of two observed paths. If it is too coarse-grained and combines routers from different ASes into a single entity, it might fail to capture routing policy.
- How do we select among the many different candidates of composed paths?
- How do we estimate the end-to-end latency once we have performed path prediction?

A summary of the various techniques we employ in addressing these challenges is listed in Table 1.

2.2 Building an Atlas

Our primary tool for building the atlas is `traceroute`, which allows us to identify the forward path from the probing entity to the destination. Traceroute also provides us with hop-by-hop round-trip times from which one could infer the latency of path segments. Asymmetry in the return routes, however, complicates this inference. We also use targeted probes to network interfaces, some of them with UDP messages and some with ICMP messages, in order to identify interfaces that are co-located at the same router or at the same PoP, so that they can be clustered together in the atlas.

We need geographically distributed vantage points in order to build the atlas. PlanetLab servers that are located at over 300 sites around the world are obvious choices for our measurement infrastructure. Probes can be issued from PlanetLab nodes at a reasonably fast rate; a traceroute probe every second translates to measurement traffic of about 4Kbps, a modest load on the PlanetLab nodes. We issue probes from the PlanetLab nodes to representative IP addresses in routable BGP prefixes, choosing just one IP address for a prefix, thereby minimizing the measurement load.

The PlanetLab measurements can be augmented with measurements from traceroute servers that outnumber PlanetLab nodes and are also more diverse in terms of administrative domains. We use these servers sparingly, issuing a traceroute request only once every few minutes. In spite of this limitation, they serve as a valuable additional source of information about local routing policies.

Finally, we use BGP snapshot information from RouteViews [11] to determine which routers belong to which autonomous systems, to infer when routes cross organizational boundaries, and to determine what subset of prefixes are to be probed by the vantage points. We leave for future work determining the improvement in prediction accuracy by using end-host data from sources such as DIMES [16].

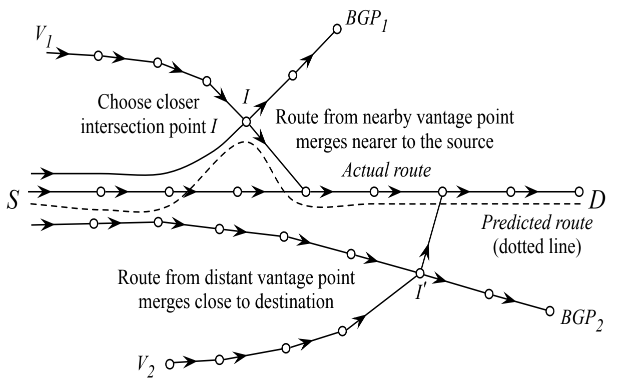
2.3 Client Integration

We envision a system where the source issues a small number (e.g., 10) of traceroute probes in order to integrate itself into the Internet atlas constructed by the vantage points. Clients issue probes to representative destinations in a small randomly chosen subset of the prefixes obtained from BGP tables.

The client can then issue queries for latency estimates of arbitrary paths. These queries will be processed using the measurements contributed by the client in combination with the atlas gathered from the vantage points. Paths from only the sources are assumed to be available, with *none from the destinations*, reflecting the usage model where a client wants to communicate with some destination that is not necessarily participating in the infrastructure. Measurements contributed by a client will be used only to service queries from that client, in order to make the system robust

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 2.2
Traceroutes from clients	Every client performs traceroutes to destinations in 10 randomly chosen prefixes	Characterize access link latency and local routing behaviour	Section 2.3
Return TTL clustering	Router interfaces that return similar TTLs to a large number of vantage points are clustered together	Cluster routers into PoPs	Section 2.6
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 2.4 and 3.1
Inference of segment latencies	Latency of an observed routing segment is estimated from the traceroutes in which the segment occurs	Predict end-to-end latencies	Section 2.5 and 3.3

Table 1: Summary of techniques employed in our approach



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

Figure 1: The route from S to D is obtained by composing a route from S with a route to D from a vantage point close to S (V_1). BGP_1 and BGP_2 are destinations in two random prefixes to which S performs traceroutes.

to misbehaving clients. These measurements will also be garbage collected by the prediction engine after the client leaves the system.

2.4 Path Prediction

Figure 1 depicts how we predict the path from a source S to a destination D using the few paths going out from S and paths from all vantage points to D. We seek to determine a path from one of our vantage points to D that has a significant overlap with the actual path from S to D. We traverse paths going out from S until we 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 ($S \rightarrow I$) with the segment from I to D ($I \rightarrow D$). In case no intersection is found, we then use paths from our vantage points to S instead of paths from S, and intersect these with paths to D. Similarly, when predicting the reverse path back from D, from which no paths are present in the system, we splice paths from our vantage points to D with paths to S.

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, we approximate the path between S and D with the path predicted between the representative end-hosts in the same prefixes as S and D. Further, we extend this predicted path to S if the few paths contributed by S intersect with the path segments to the representative end-host in the same prefix as S.

There are of course many pairs of path segments that can be composed to predict the end-to-end path. Which path should one pick? We approximate the path the Internet chooses by modeling how Internet routing works in practice. We 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, interdomain routing prefers shorter AS paths over longer ones in the absence of conflicting locally preferred policies, and intradomain routing uses latency of paths within the AS to determine the appropriate exit point. In Section 3, we show how to use these metrics to determine which combination of path segments to compose to determine the end-to-end path.

2.5 Latency Estimation

We estimate the round-trip latency by a simple technique that considers the paths predicted for the forward and reverse directions. For the forward direction, we first generate an estimate for the delay from S to I by halving the RTT measured to I as part of the traceroute from S. When using a path measured from a vantage point to S, we estimate the delay from S to I as half the difference of the RTTs reported by probes to S and I. We similarly estimate the delay from I to D. These do not necessarily represent the true latencies of segments ($S \rightarrow I$) and ($I \rightarrow D$) since the reverse path back from any hop observed on a traceroute could be asymmetric. So, we approximate the true latency of a segment as the median of all its latency estimates. For example, if the segment ($I \rightarrow D$) is seen on traceroutes to D from V_1, V_7, and V_9, we derive an estimate for this segment from each of these traceroutes (the difference between RTTs from V_i to D and from V_i to I) and then approximate the latency along this segment as the median of these estimates. Since we typically observe a segment from multiple vantage points, the hope is that the median is close to the true latency. We sum latencies over segments instead of links to prevent accumulation of errors associated with the estimate for each link. The estimated latency of the forward path is simply the sum of the delay estimates for the two segments ($S \rightarrow I$), and ($I \rightarrow D$). We similarly estimate the reverse path latency. The estimated round trip time is the sum of the estimated latencies for the forward and reverse paths.

2.6 Clustering Routers

Our approach allows path compositions to be determined at various granularities. One could declare two paths to have intersected only if the same network interface address appears on both paths, but such a strategy would fail to compose paths that pass through distinct network interfaces co-located on the same router/PoP. On the other hand, declaring an intersection if both paths traverse a common AS is prone to significant prediction error as it allows for composition of paths passing through different PoPs within the same AS. Thus, paths need to composed at a level coarser-grained than network interfaces but finer-grained than ASes.

We determine clusters of router interfaces 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, we only cluster routers that belong to the same AS. There is a tradeoff between the utility of clusters and their correctness. Increasing cluster sizes initially improves prediction accuracy by detecting missed intersections, but subsequently degrades accuracy by predicting non-existent paths. Our clustering algorithm attempts to hit the knee of this curve.

Our clustering methodology involves two distinct steps — alias resolution and intra-AS clustering. For clustering router interfaces into routers by resolving aliases, we use the source-address based technique employed by Mercator [4]. Next, we cluster routers into PoPs by probing each router from a large number of vantage points. We use the response TTL value to estimate the length of the reverse path, as done in [10]. Routers in the same AS that are geographically nearby will normally take similar reverse paths back to the vantage point. We associate each router with a *reverse path length vector* — a vector with as many components as the number of vantage points, in which the i^{th} component is the length of the reverse path from the router back to the i^{th} vantage point. If two path length vectors are similar, in the sense that the average and the maximum of the component-wise differences are below certain thresholds, then the two corresponding routers are grouped together in the same cluster.

Of the 341,602 router interfaces that we observed in all our traceroutes, 260,637 responded to our ICMP probes from at least half of the 143 PlanetLab nodes acting as vantage points, and these were clustered into 62,453 clusters. Improvements to our clustering algorithm using geographic information based on DNS names [19, 14] are presented in [8].

3. EVALUATION

We use nodes in PlanetLab, and several public traceroute and looking glass servers to perform our measurements. Our atlas comprises measurements to a wide range of destinations made from 158 PlanetLab nodes that serve as our vantage points. We obtained the list of all globally routable prefixes from RouteViews [11] and in each prefix, determined a *.1* address that responded to probes. On 3 February 2006, we performed traceroutes from our vantage points to 87,334 destinations. Our atlas also includes traceroutes performed on the same day from 582 public traceroute and looking glass servers to destinations in 200 random prefixes each. To evaluate our model, we use paths measured from these traceroute servers as our validation set. Whenever we use a traceroute server as a client, we exclude measurements made from it from the atlas. Instead we include what it would do as a typical client; a few traceroutes to random prefixes, not including the test cases.

3.1 Does our atlas include the true AS path?

We begin by evaluating the feasibility of AS path prediction. To compare the actual and predicted AS paths, we define an AS path similarity metric that is similar to the RSIM metric used in [6]. We 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

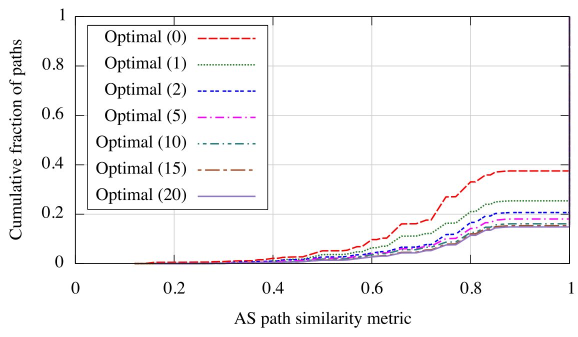


Figure 2: Accuracy of AS path prediction with varying number of paths from the source. The number of paths available from each source is in brackets.

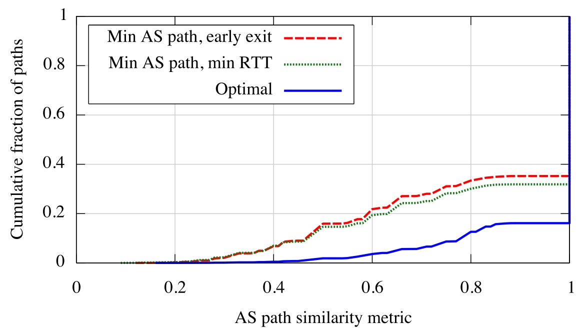


Figure 3: Comparison of AS path prediction accuracy across policies used for predicting paths.

of ASes in each of the paths; the ordering of ASs in the paths is not considered. The maximum value of this metric is 1, when both paths pass through exactly the same set of ASes.

For each source-destination pair, we considered all intersections obtained using the basic technique of intersecting paths from/to the source with paths to the destination in our atlas. We refer to the policy that chooses (using the validation data set) the intersection such that the resulting predicted path maximizes the AS similarity metric as the *optimal policy*. Figure 2 plots the distribution of the similarity metric for this optimal policy. Availability of 10 paths from the source to random destinations increases the fraction of paths for which the optimal policy gets the AS path exactly right from 63% to 84%. We found 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 our evaluation, we assume that we have 10 paths from each source.

3.2 Can we predict the AS path?

The previous result demonstrates that in most cases, the actual AS path exists in our atlas as a combination of path segments. To find this path, we need to determine a policy to choose among the set of path segments joining the source and destination. We evaluated a handful of policies. The policies that worked best, both in terms of predicting paths and estimating latencies, choose I (the intersection) that minimizes end-to-end AS path length along the predicted path. Minimizing AS path length approximates BGP's default objective function, and hence, is likely to choose a policy compliant path. Among those that minimize the AS path length, we consider two policies. The first chooses I such that the distance to exit the first-hop AS is minimized. Minimizing the distance, in

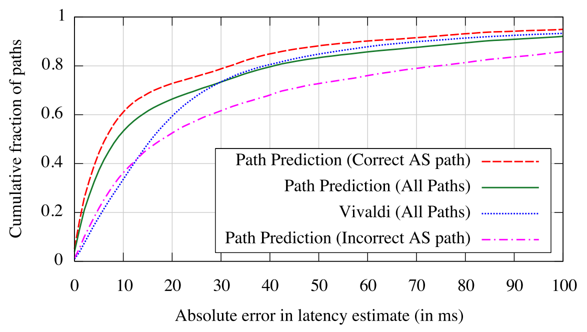


Figure 4: Comparison of latency estimates based on our predicted paths with Vivaldi based estimates.

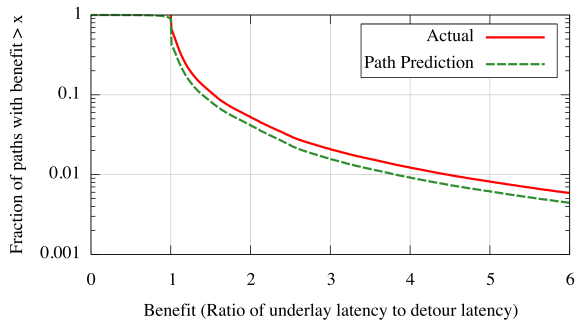


Figure 5: Benefits of choosing detours using predicted latencies. Note that the graph has a log-scale y axis.

terms of latency, covered by the path within the first-hop AS encodes the default early exit intradomain routing policy. The second picks the I closest in RTT to the source. We refer to these two policies as *(Min AS path, early exit)* and *(Min AS path, min RTT)*. Figure 3 plots the accuracy with which these policies help in finding the correct AS path, for the case when we have 10 traceroutes from the source. Among the 83% of paths for which the AS path exists in our atlas, either policy predicts the AS path correctly for at least 65% of paths.

Since our model does not involve use of inferred AS relationships, the paths we predict could violate policy routing. To see if this is a problem, we applied Gao's AS relationship inference algorithm [3] on AS paths observed in all our measurements as well as on AS paths obtained from the RouteViews [11] BGP dump. We then computed how many of the AS paths in our validation set, and how many of the predicted AS paths, violate valley-free routing [3]. Of the 42,028 paths in our 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 our policies do usually manage to find policy-compliant paths. As noted earlier, our choice of an intersection close to the source minimizes the number of policy decisions our model must predict. This result also suggests that constraining our model to only choose amongst policy-compliant candidate paths will not decrease its predictive ability.

3.3 Can we estimate latencies?

We now study the accuracy of our latency estimates in comparison with that of one of the best existing coordinate-based systems, Vivaldi [1]. Feeding the latencies of all paths in our atlas into Vivaldi, we generate 2-dimensional Euclidean coordinates with height vectors for all end-hosts in our validation set. Vivalidi was run until its coordinates converged. We use these coordinates to predict latencies. The number of measurements used in our approach is comparable with that given as input to Vivaldi, but uses traceroutes instead of pings. Our predictions are based on the paths traced by traceroutes from the vantage points to various prefixes, while Vivaldi's coordinates were generated from the end-to-end latencies for all of these measured paths.

Figure 4 compares the latency estimates obtained using Vivaldi with the estimates obtained using our predicted paths. Our latency estimates, obtained using 10 paths from the source, are better than those yielded by Vivaldi's coordinates in terms of absolute error. For example, 54% of our predictions across all paths are within 10 ms of actual latency, while only 35% of Vivaldi's are. Our latency estimates are also better than those of Vivaldi in terms of relative error, the metric for evaluation used in [1].[1] The latency estimates shown in Figure 4 were those obtained using the *(Min AS path, min RTT)* policy; the accuracy of estimates obtained with the *(Min AS path, early exit)* policy were similar.

Our approach of estimating latency based on predicted paths also helps us reason as to *why* an estimate is accurate/inaccurate. Figure 4 shows the accuracy of our latency estimates for the cases when we predict the AS path correctly, and when we do not. Our latency estimates are significantly more accurate when the predicted AS path is correct.

3.4 Can we predict detours?

Detecting better latency detours is one of the several applications enabled by our approach. Note that Vivaldi, by construction, predicts no detours. To evaluate how well our latency estimates preserve detours, we consider only paths measured from PlanetLab nodes. From the 158 PlanetLab nodes in our dataset, we chose 35 at random to serve as the sources in our experiment. The remaining 123 PlanetLab nodes serve as potential detour nodes. For each path from the 35 sources, we compute the best latency path, including the direct path to the destination and the set of one-hop detour paths via any of the 123 detour nodes. We then do the same using the predicted latency for the direct path to the destination. Figure 5 compares the detour benefits (ratio of underlay to detour latency) obtained. Our model predicts the existence of a detour for 43% of all paths whereas 66% actually show a benefit. Though we predict a detour when it does not exist for 10% of paths, only for 1% of all paths does the chosen detour stretch the latency of the path by more than 30% (not shown in the figure).

4. RELATED WORK

This work provides a bridge between two islands of research: Internet measurement and Internet distance estimation.

Several recent measurement studies analyze the effects of Internet structure and routing policy on path performance. Spring et al. [18] use an analytical model of routing to isolate different factors that cause a path through the network to be longer than necessary. Our results are consistent with theirs in that modeling the AS path is important for accuracy of latency prediction. Spring et al. choose a restricted subset of the Internet to study and use geographic path distance as a baseline to analyze latency inflation. Our goals are different: we seek to find the simplest composition of internal network performance information and topology that can be used to predict Internet path latency.

[1]In our experiments, the relative error distribution for Vivaldi is consistent with that reported in [1].

Internet distance estimation has been motivated by applications such as overlay routing networks and server selection, e.g., a recent Bittorrent client, Azureus, uses Vivaldi [1] to select nearby peers. Such applications have spurred a large body of black box techniques to predict latencies, which broadly fall into two broad categories, the *landmark* approach and the *dimensionality reduction* approach.

Landmark Approach IDMaps, one of the first techniques to predict Internet latencies, estimates the latency between two clients i and j as the latency from i to its closest vantage point plus the latency from j to its closest vantage point plus the latency between the two vantage points. Our approach can be seen as a generalization of IDMaps, by using measured information about the structure of the network to yield better composite paths. King [5] is similar to IDMaps but uses DNS servers as landmarks. Meridian [20] performs on-demand probing from a carefully chosen subset of its landmarks to determine the landmark closest to a given destination. Meridian implicitly assumes an underlying metric space to predict the closest node, and it is unclear how to extend it to predict latency between an arbitrary pair of nodes.

Dimensionality Reduction Approach The dimensionality reduction approach attempts to compactly represent measured latencies between vantage points and clients in a low dimensional geometric space. The techniques in this space can be classified as *metric*, i.e., the resulting distances satisfy the triangle inequality, and nonmetric.

- Metric Embeddings: GNP [12] pioneered the dimensionality reduction approach by embedding vantage points and clients in a low-dimensional Euclidean space. Vivaldi [1], the model we use for comparison, uses a non-Euclidean embedding where nodes lie on a plane with a positive height coordinate and the distance between a pair of nodes is the sum of their heights plus the distance along the plane. Earlier published data shows that these blackbox techniques outperform both IDMaps and King [1, 12]. However, a serious drawback of metric embeddings [7, 21] is that the method is fundamentally incapable of detecting detours, because Euclidean and Vivaldi-like distances obey the triangle inequality.

- Nonmetric Approaches: Shavitt and Tankel [17] embed nodes in a hyperbolic coordinate space and demonstrate improved accuracy over Euclidean embeddings. However, the hyperbolic coordinate system has been shown to perform poorly compared to Vivaldi [7], the coordinate system against which we evaluate our model. More recently, Mao et al. [9] proposed a dimensionality reduction technique based on matrix factorization. Explicitly accounting for the asymmetry of paths and using a nonmetric embedding makes this approach promising. However, it is unclear how to obtain one-way latencies from vantage points to arbitrary clients; the evaluation in [9] assumes symmetric RTTs.

5. CONCLUSION

In this paper, we have presented a simple structural approach for predicting Internet path latencies that outperforms black box techniques. Our latency predictions are based on an underlying path prediction model that can predict PoP-level paths with high accuracy. Encouraged by the results obtained, we have extended our approach to predict other path properties such as bottleneck capacity and loss rate, as part of the *iPlane* system [8]. We plan to make *iPlane* publicly available as a service in the near future.

Acknowledgments

We would like to thank members of the DIMES project [16] for providing us an independently collected data set for validation and Morley Mao for the AS path and relationship data set used in our evaluation. This research was partially supported by the National Science Foundation under Grants CNS-0435065 and CNS-0519696.

6. REFERENCES

[1] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, 2004.
[2] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global Internet host distance estimation service. *IEEE/ACM Trans. on Networking*, 2001.
[3] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. on Networking*, 2001.
[4] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *INFOCOM*, 2000.
[5] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *IMW*, 2002.
[6] N. Hu and P. Steenkiste. Quantifying Internet end-to-end route similarity. In *PAM*, 2006.
[7] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of embeddings for Internet coordinate systems. In *IMC*, 2005.
[8] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.
[9] Y. Mao and L. Saul. Modeling distances in large-scale networks by matrix factorization. In *IMC*, 2004.
[10] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level path inference. In *SIGMETRICS*, 2005.
[11] D. Meyer. Routeviews. `http://www.routeviews.org`.
[12] E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.
[13] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. on Networking*, 1997.
[14] Sarangworld project. `http://www.sarangworld.com/TRACEROUTE/`.
[15] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: A case for informed Internet routing and transport. *IEEE Micro*, 19(1), 1999.
[16] Y. Shavitt and E. Shir. Dimes: Let the Internet measure itself. *CCR*, 35(5), 2005.
[17] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *INFOCOM*, 2004.
[18] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *SIGCOMM*, 2003.
[19] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Trans. on Networking*, 2004.
[20] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *SIGCOMM*, 2005.
[21] H. Zheng, E. K. Lua, M. Pias, and T. Griffin. Internet routing policies and round-trip-times. In *PAM*, 2005.