# Path Stitching: Scalable and Systematic Internet-Wide Path and Delay Estimation from Existing Measurements

DK Lee* KAIST
Keon Jang* KAIST
Changhyun Lee* KAIST

Gianluca Iannaccone† Intel Research Berkeley

Sue Moon* KAIST

CS/TR-2009-313

July 24, 2009

K A I S T
Department of Computer Science

* {dklee, keonjang, chlee, sbmoon}@an.kaist.ac.kr

† gianluca.iannaccone@intel.com

# Path Stitching: Scalable and Systematic Internet-Wide Path and Delay Estimation from Existing Measurements

[CS/TR-2009-313]

DK Lee, Keon Jang, Changhyun Lee, Gianluca Iannaccone†, and Sue Moon
Computer Science Department, KAIST
{dklee, keonjang, chlee, sbmoon}@an.kaist.ac.kr
† Intel Research, Berkeley
gianluca.iannaccone@intel.com

## ABSTRACT

Internet-wide services and applications depend on accurate information about the internal network state to deliver good performance to end-users. However, today's Internet does not provide such information explicitly and a number of systems have been recently proposed and implemented to provide a shared measurement infrastructure for distributed applications [1–4]. The goal of this work is to demonstrate that *without any new* measurement infrastructure or active probing we obtain composite performance estimates from AS-by-AS segments and the estimates are as good as (or even better than) those from existing estimation methodologies that use on-demand, customized active probing. The key idea behind scaling measurements to the size of the Internet is to take advantage of the known underlying structure of the network.

The main contribution of this paper is an estimation algorithm that breaks down measurement data into segments, efficiently identifies relevant segments, and by carefully stitching segments together, produces delay and path estimates between any two end points. Fittingly, we call our algorithm *path stitching*. Our results show remarkably good accuracy: error in delay is below 20 ms in 80% of end-to-end paths. We show also that our path stitching approach performs comparably to existing iPlane without having to instrument any new measurement node.

## 1. INTRODUCTION

Internet-wide services and applications depend on accurate information about the internal network state to deliver good performance to end-users. For example, content distribution networks use path or delay information to direct clients to replicas that would provide best performance. Peer-to-peer VoIP systems (e.g., Skype) have been shown to deliver better voice quality when AS path information is taken into account for selecting peers [5]. However, today's Internet does not provide such information explicitly and developers resort to ad-hoc measurement tools to obtain the necessary data. This poses an additional tax on the development cost of new services and applications. For this reason, a number of systems have been recently proposed and implemented to provide a shared measurement infrastructure for distributed applications [1–4]. They follow a common plan of action: $(1)$ define estimation methodologies for delay, path, loss rates, etc.; $(2)$ carefully construct an active probing strategy and instrument end-systems to collect measurements accordingly.

In this work we diverge from this tradition of active measurement. We are interested in the potential of estimation methodologies in isolation from data collection. Our goal is to demonstrate that *without any new* measurement infrastructure or active probing we obtain composite performance estimates from AS-by-AS segments and the estimates are as good as (or even better than) those from existing estimation methodologies that use on-demand, customized active probing. The key idea behind scaling measurements to the size of the Internet is to take advantage of the known underlying structure of the network. Existing approaches in line with ours are iPlane [1] and Akamai's core points [6]. They derive estimates by composing performance measures of network segments along the end-to-end path. Our approach differs from these two in that we construct end-to-end information from performance measures segmented *by the AS*. Let us illustrate our approach in the following simple example. Consider a query for some performance metric between two points $x$ and $y$ in the Internet. Assuming that we have ac-

cess to segmented performance measures, we infer the AS path between the two points and construct the end-to-end metric corresponding to those ASes on the inferred path. Our AS-based structural approach to Internet-wide end-to-end path and delay estimation obliterates the triangular inequality problem in network embedding [7].

The main contribution of this paper is an estimation algorithm that breaks down measurement data into segments, efficiently identifies relevant segments, and by carefully stitching segments together, produces delay and path estimates between any two end points. Fittingly, we call our algorithm *path stitching*. In this work we use *path* and *round-trip delay* as measures of interest for validating our structural approach.

Our approach is based on the following assumptions: (1) AS-level path inference is accurate; (2) measurement data segmented by the AS is readily available; and (3) characteristics of end-to-end path retain temporal stability. We argue that these assumptions are reasonable and present surmountable challenges. AS path inference has been an active area of research [8–10] and published methodologies now report 90% accuracy in AS path inference. Numerous end-to-end measurement data sets are publicly available today [1,11,12] that we can utilize. Finally, with regard to the third assumption, routes from a single end-host to many destinations are known to be fairly stable despite Internet's inherent dynamic nature [13, 14].

Our results show remarkably good accuracy: error in delay is below 20 ms in 80% of end-to-end paths. We also present a comparison with iPlane [1], where measurements are carefully designed from hundreds of vantage points to maximize accuracy. Path stitching show an accuracy similar or slightly better than iPlane without having to instrument any new measurement node.

The accuracy of our path stitching algorithm defines what is *already* achievable without instrumenting or deploying any additional measurement node. We believe it is about time that we look into the possibility of taking the most out of existing measurements and their infrastructures, instead of building yet another measurement system. Our work is a step towards bringing a diverse set of measurements together to improve accuracy without additional active measurements. Path stitching represents a reference baseline valuable when it comes to understand the benefits (and the costs) of deploying a new measurement infrastructure.

The rest of the paper is organized as follows. The next section describes data sets that will be used throughout the paper. Section 3 presents the path stitching algorithm. Our approach of segmenting existing measurements and stitching them up may lead to two complications: there may not be any valid path between a pair of end systems and we need to resort to approximation methods (Section 4); or there may be too many solutions and we need to trim down the list of candidate paths (Section 5). Also, we must assume that the source data sets contain inherent measurement errors. Our algorithm avoids amplifying them by implementing mitigation techniques (Section 6). Section 7 presents an evaluation of each step of the algorithm as well as a comparison with existing path and round-trip delay estimation approaches. Finally, Section 8 discusses the related work that led us to this work and Section 9 summarizes our conclusions and presents future work.

## 2. DATA

We use three measurement data sets from three independent sources: Routeviews [15] and RIPE [16] BGP routing table snapshots and Ark [12] traceroute outputs. These datasets are among the largest data archives publicly available and hold constantly updated information about IP and AS-level topologies. Thus they provide a good starting point for our investigation into the feasibility of path stitching.

CAIDA's Ark project collects traceroutes from 18 monitors to every /24 routable prefix. From Ark, we use one round (cycle-20080407) of traceroute outputs taken from April 7th to 9th, 2008 (a total of approximately 14 million traceroute outputs).

RouteViews and RIPE are two most widely used repositories of BGP routing table snapshots. We use BGP routing table snapshots from RouteViews' BGP listener, route-views.oregon-ix.net. We also use snapshots from RIPE NCC Routing Information Service (RIS). RIS operates 14 monitoring points (rrc00 - rc07 and rrc10 - rrc15) and each monitoring BGP listener peers with different ASes. Both snapshots are from the same three-day-period as our Ark data.

### 2.1 Evaluation Data Set

To evaluate the accuracy of our estimation methodology, we take direct path and delay measurements between a set of hosts and compare them against the path stitching estimates. We ran traceroute 50 times a day between 184 PlanetLab (PL) nodes during the same period as the Ark data. For 569 pairs we failed to collect any path or delay measurements because of long-lasting host failures and excluded those pairs from our data set. We also discard pairs for which traceroutes do not reach the destinations. The row `planetlab` in Table 1 summarizes the AS-level characteristics of the data set. Numbers in ()'s represent ASes that appear in the PL data set but are not covered in the Ark data set. A total of 6 stub ASes are not seen by any of the Ark monitors.

| | # pairs | # ASes | Transit AS | Stub AS |
|---|---|---|---|---|
| `planetlab` | 10,539 | 276 (6) | 197 (0) | 79 (6) |
| `pl-easy` | 462 | 199 (4) | 129 (0) | 70 (4) |
| `pl-hard` | 10,077 | 270 (5) | 194 (0) | 76 (5) |

**Table 1: AS-level statistics for PL nodes.**

We then split this data set in two smaller sets: `pl-easy` and `pl-hard`. The distinction between the two sets lies in the co-location of Ark monitors at the source AS. Pairs in the

pl-easy set have the source node located in the same AS as an Ark monitor (namely, *amw-us, cbg-uk, cjj-kr, dub-ie*, and *gig-br*). There are 462 pairs in the set. We expect that path stitching to return the most accurate results on this set of nodes. The latter set pl-hard contains the remaining pairs whose source PL nodes are not in the same ASes as any Ark monitor. This set is useful to evaluate our methodology for uncharted (or partially measured) network regions[1].

# 3. PATH STITCHING

Our goal is to estimate end-to-end path and round-trip delay between any two hosts in the Internet without resorting to active probes but re-using existing network measurement data. In the design of the path stitching algorithm we are guided by two main objectives: (i) coverage: the algorithm must be able to answer even those queries about end systems that are not present in the existing measurement data sets; and (ii) accuracy: the estimate should be as close to the actual measurement as possible. As a first step, we focus on path and round-trip delay as they address two basic performance measurement needs for any distributed application. From an application developer's standpoint, path stitching appears as a simple query-based Internet service: a query consists of two IP addresses (source and destination) and the results contain a ranked list of candidate router-level paths and round-trip delay estimates. Now we begin the description of the path stitching algorithm.

## 3.1 Algorithm

Given two IP addresses as input, the path stitching algorithm operates as follows:

**Step 1: Map IP addresses to AS numbers.** We use the BGP routing tables to map an IP address to an AS number. The longest prefix match on the IP address returns the prefix and corresponding AS path. The last AS number is then taken as the origin AS for the host. Section 6.1 describes how we handle the case when the mapping is incorrect or when a single IP address maps to multiple origin ASes.

**Step 2: Infer AS-level paths between ASes.** We follow KnownPath's methodology [9] to infer AS paths between two ASes. KnownPath exploits the AS paths already present in BGP routing tables and infers AS paths by extending these known AS paths. Its inferred AS paths always conform to the valley-free property of AS paths [18]. In Section 6.2 we extend KnownPath to improve its accuracy.

**Step 3: Stitch path segments along the inferred AS path.** Taking as input the inferred AS path from Step 2, we extract router-level path segments from the traceroute database and stitch them up along the inferred AS path. This step may result in no candidate paths or may lead to too many candidate paths. We discuss approximation methodologies for the former case in Section 4 and preference rules for the latter in Section 5.

**Step 4: Return the best candidate paths and delays.** When Step 3 outputs stitched paths, the final step is to calculate round-trip delays along the paths and return them as query result. The results contains both the most recently measured round-trip delays as well as a distributions of all the measured delays along the path.

Each step above makes use of the Ark and BGP data. In order to handle potentially large data sets efficiently, we preprocess and convert them to a more easily manageable format. Next, we describe the data conversion process.

## 3.2 Constructing the path segment repository

We split traceroute outputs into intra-domain segments and inter-domain segments. The set of intra-domain segments of a AS $A$ (indicated by $:A:$) cover all known paths between any ingress and egress points of the AS $A$ (together with router-level and latency information). The set of inter-domain segments between AS $A$ and AS $B$ (indicated by $A::B$) describe all inter-AS connections that appear in the Ark data set. These "path segments" represent the basic components that are later stitched by our algorithm. Table 2 contains sample path segments from two traceroute outputs, one from host $a$ to host $b$, $\langle a, a_1, a_2, b_1, b_2, b\rangle$ and the other from host $a'$ to $b'$, $\langle a', a_1, a_3, b_3, b_2, b'\rangle$. All $a$ addresses belong to AS $A$ and all $b$ addresses belong to AS $B$. The actual size of the repository constructed from 14 million traceroute outputs of the Ark data set is: 2.1M intra-domain segments and 0.6M inter-domain segments. About 1M unique IP addresses are observed.

| Segment | Ingress | Egress | Interim | Delay |
|---|---|---|---|---|
| $:A:$ | $a$ | $a_2$ | $a_1$ | $d_A$ |
| | $a'$ | $a_3$ | $a_1$ | $d'_A$ |
| $:B:$ | $b_1$ | $b$ | $b_2$ | $d_B$ |
| | $b_3$ | $b'$ | $b_2$ | $d'_B$ |
| $A::B$ | $a_2$ | $b_1$ | – | $d_{AB}$ |
| | $a_3$ | $b_3$ | – | $d'_{AB}$ |

**Table 2: Path Segment Repository Example**

## 3.3 Query Example

We use the repository in Table 2 to illustrate how we resolve a simple query from host $a$ to host $b$. First, we use the BGP routing tables to map $a$ to AS $A$ and $b$ to AS $B$ (**Step 1**). Again from the BGP tables we derive the AS-level path from $a$ to $b$ (**Step 2**), $AB$ in this example. Then, among the intra-domain segments of $A$, we search for those that start with $a$ and find $\langle a, a_1, a_2\rangle$. Then we resolve the portion $A::B$ by looking for the same ingress and egress points in

[1]The inter-domain connectivity of Planetlab nodes relies on research networks and may not be representative of the Internet [17]. However, the goal of this data set is to evaluate the accuracy of our path stitching. We leave the problem of representativeness for future work.

the repository and find a segment $\langle a_2, b_1 \rangle$ that rendezvouses with the $:A:$ segment at $a_2$. We continue to stitch up in a similar manner, and the final query result is one single path: $\langle a, a_1, a_2, b_1, b_2, b \rangle$ with delay $d_A + d_{AB} + d_B$.

Actual queries are more complicated to answer than the above sample case. The following sections address the challenges in detail.

# 4. APPROXIMATION IN PATH STITCHING

Path stitching does not always return a stitched path. It fails when the path segment repository is missing data for the following three reasons: $(i)$ the source or destination IP address maps to an AS that is not present in the repository; $(ii)$ the inter-domain segment is not present in the repository; or $(iii)$ the end of a path segment does not match any of the beginnings of the next path segments (i.e., the segment cannot be stitched).

| Data Type | Total AS | Transit AS | Stub AS |
|---|---|---|---|
| Ark | 14378 | 4418 | 9960 |
| BGP | 28244 | 4847 | 23397 |

**Table 3: Number of ASes in Ark and BGP data**

The first case of missing ASes has no solution other than collecting more measurements. Ark monitors operate in a coordinated manner to probe every routable /24 prefix. Yet the Ark dataset is missing 50% of ASes present in the BGP tables (see Table 3). A careful look at the data reveals that the Ark dataset covers 93% of the transit ASes and just 42% of the stub ASes[2]. Furthermore, 89% of the missing ASes ($12,382$ out of $13,866$) correspond to traceroutes that did not reach their intended destinations and returned incomplete results. The $13,866$ ASes not covered in the Ark data set correspond to approximately 110 million, or 5.8%, of the IP addresses that originate from all ASes in the BGP data: an order of magnitude smaller percentage than in the number of ASes. For those IP addresses, our path stitching cannot generate any estimate. For future work, we consider incorporating a second traceroute-based dataset (DIMES [19]) to increase the IP address coverage above the current $94.2\%$ and AS coverage from $50.9\%$.

The second case of missing inter-domain segments is common. A possible work-around is to search for a reverse segment. That is, if we cannot find an inter-domain segment $A::B$, we consider the reverse path segment, $B::A$, instead, as a potential candidate. This is reasonable given that inter-domain segments are typically over point-to-point links and all links are bi-directional[3]. Our data set contains $61,606$ sets of inter-domain segments and the number grows to $103,030$ after we incorporate the reverse inter-domain segments.

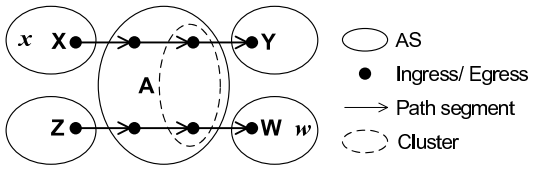



**Figure 1: Example of clustering**

We address the third case by clustering IP addresses. Figure 1 illustrates our solution. In the example, the Ark data set contains the AS paths $X::A::Y$ and $Z::A::W$. When a query for IP addresses $x$ and $w$ arrives, the algorithm infers correctly an AS path $X::A::W$, but it is not able to find in the repository segments that can be stitched together. As in the case of inter-domain segments, we reverse intra-domain segments and see if the end points of the segments line up. Reversing intra-domain segments is in line with the way network operators set link weights for computing the shortest paths, i.e., the same weight in both directions [20, 21]. In this example reverse intra-domain segments do not help. We then resort to approximation by clustering: the dotted circle in Figure 1 collapses two egress points into one. We employ multiple levels of clustering, first by the router (called IP aliasing), the Point-of-Presence (PoP), and the prefix length. Clustering at the router level has been addressed in previous work [22, 23] and there exist data sets that have resolved IP aliasing. Recent work by Madhyastha *et al.* offers router aliases as well as PoP clusters [1]. In our evaluation we use their router aliases and PoP clusters from January 2008 and March 2008, respectively. We extend approximation one step further than in [1] and allow clustering by the prefix if router or PoP clustering fails.

The first case is only addressable with additional data. In Section 7 we evaluate the approximation techniques only for the last two categories.

# 5. PREFERENCE RULES

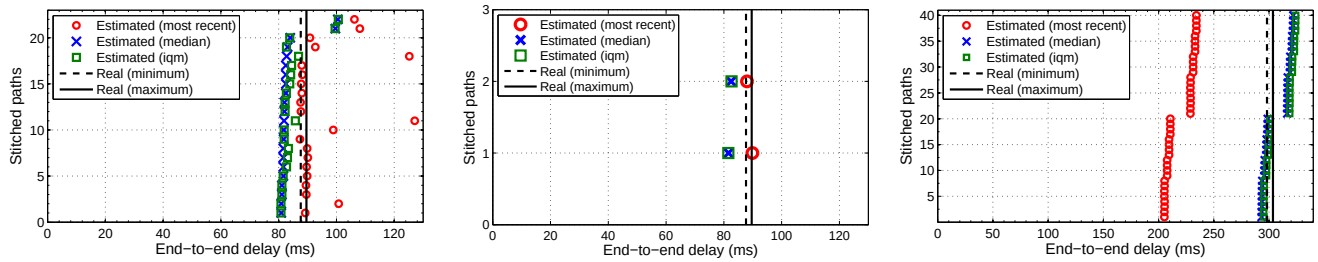
In the previous section, we have discussed cases where there are no stitched paths after **Step 3**. Now, we turn our attention to the cases where there are *too many* stitched paths for a given query. Our goal in this section is to define the rules of preference and apply them to trim the list of candidate paths. A key insight in designing good preference rules is to reflect the actual mechanism that route packets through the network. A good preference rule is therefore one that effectively discriminates among the various stitched paths without compromising the accuracy of delay estimates.

## 5.1 Finding clues to preference rules

To get a sense of the challenges ahead, we take a quick look at a few PlanetLab measurements described in Sec-

[2]We call an AS transit if it appears in any AS path (not the first nor the last AS) in a BGP table; otherwise, stub.

[3] Note that early exit routing policies (often called hot-potato routing) only affect intra-domain paths, not AS-level paths.




(a) planetlab1.csail.mit.edu → planet2.scs.stanford.edu

(b) filtering planetlab1.csail.mit.edu → planet2.scs.stanford.edu

(c) planetlab2.xeno.cl.cam.ac.uk → pl1-higashi.ics.es.osaka-u.ac.jp

**Figure 2: Two examples to demonstrate differences between stitched paths**

tion 2.1. We examine the stitched paths and their differences and scout for clues on how to discriminate among them. In Figure 2 we compare estimated delays from stitched paths with on-the-spot measurements for two representative pairs of PlanetLab nodes.

The first pair (*mit.edu* to *stanford.edu*) comes from PlanetLab nodes that are located in ASes different from any Ark monitor. **Step 3** of the path stitching algorithm returns 22 candidate paths. The paths traverse two ASes and are derived by stitching together one segment in the source AS, 11 segments in the destination AS (AS32 – Stanford University) and 2 inter-domain segments. Figure 2(a) plots the median and interquartile mean delays as well as the most recent delay estimate for each path (we may have multiple measurements per segment and thus stitched path). The dotted and solid vertical lines represent the minimum and maximum delays from 150 traceroutes we have collected (on-the-spot measurements).

Taking a second look at the original path segments, we notice that the main difference between the various candidate paths is the distance between the source and destination addresses and the start and end points of the stitched paths[4]. Only one of the 11 intra-domain segments in AS 32 belongs to the same /22 prefix as the destination address. Figure 2(b) plots the delay estimates of the resulting 2 stitched paths (we keep both inter-domain segments). The estimates, either most recent or median, are very close to on-the-spot PlanetLab node measurements. It appears therefore that restricting ourselves to paths closer to the destination (or source) address allows to drop a large number of candidate paths with little risk of degrading the estimate accuracy.

The second pair (Figure 2(c)) is between PlanetLab nodes where the source node (*planetlab2.xeno.cl.cam.ac.uk*) is located in the same AS as an Ark monitor, *cbg-uk*, and the destination (*p11-higashi.ics.es.osaka-u.ac.jp*) is not. The inferred AS path of this pair is 786 20965 2907 4730 and we obtain 40 stitched paths for this query. We plot again the most recent, median, and interquartile mean of estimated delays per path in Figure 2(c).

In this example, the estimated delays using the most recent delays are about 100 ms smaller than on-the-spot measurements, while the median and interquartile mean delays match the on-the-spot measurement for 50% of the paths. The other 50% of the paths are 20 ms larger delays. We have examined the 40 paths manually and found two intra-domain segments in AS 20965. These two path segments start and end at the same ingress and egress routers of AS 20965, but have different number of internal hops and delays; thus the overall difference of about 20 ms in Figure 2(c).

Between the two intra-domain segments in AS 20965, the segment with smaller delay is originally from a traceroute destined to a prefix in AS 4730, while the longer delays come from a path to AS 19401 (that does not belong to the inferred AS path). This example has given us a second idea for a preference rule: use the original destination prefix of the traceroute measurement to discard candidate segments.

From the above two examples, we glean two ideas for preference rules: $(i)$ *proximity* to the source and destination address of the query and $(ii)$ *destination-bound segments* that are derived from traceroutes with the same destination prefix. In the following, we examine in detail how effective these rules are at narrowing down the list of candidate paths.

## 5.2 Preference Rule #1: Proximity

IP addresses in the Internet far outnumbers ASes and there is no public data set that contains all the IP addresses. Thus end points of a query are likely to be not found in the path segment repository. Our first rule of preference addresses this problem by proximity. The proximity rule dictates that the path segments closest to the queried IP addresses are chosen for path stitching. The proximity is measured by the common prefix length. Although this rule is very effective at narrowing down the list of candidates[5], it is not clear what impact it might have on the delay estimate.

To address this concern, we plot in Figure 3 the cumu-

[4]We define "distance" as the size of the prefix that contains both addresses. For example, two addresses that belong to the same /24 prefix are "closer" than two address that belong to the same /23 prefix (but different /24 prefix).

[5]The size of the network prefix (e.g., /22) as a distance metric is an effective means of discriminating paths, as it allows strict ranking.

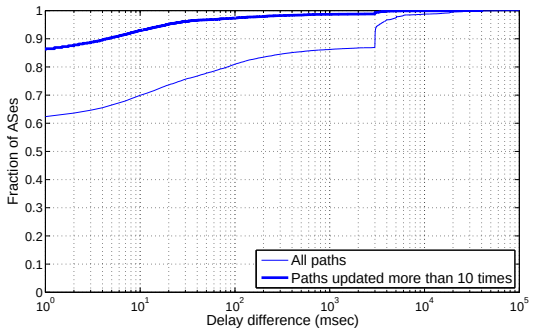



**Figure 3: Delay difference between the shortest and longest paths in an AS.**

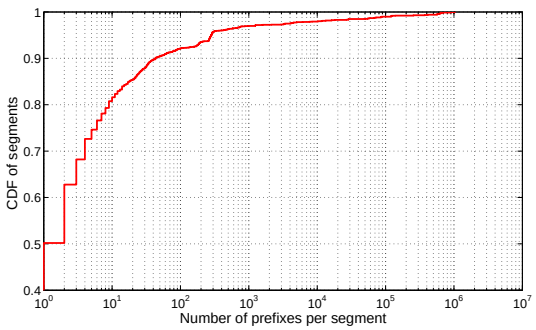



**Figure 4: CDF of number of prefixes per segment**

lative distribution function (CDF) of the difference between minimum and maximum delays across path segments within each AS in our dataset. For a path segment for which we have multiple measurements we use the median delay. The number of ASes plotted is $10,368$, that is $4,010$ fewer than the total in our Ark dataset: the excluded $4,010$ ASes are all stub ASes and have just one path segment.

About $61\%$ of the ASes included in the plot have differences smaller than $1$ ms. Between $1$ ms to $100$ ms, the number of ASes grows steadily from $61\%$ to $82\%$, and then beyond $100$ ms, there is a noticeable jump at $3$s (due to the default traceroute timeout).

In $18\%$ of ASes, the delay difference within an AS is greater than $100$ ms. For those ASes (for delay within an AS varies widely), finding the closest point to the queried host along the path segments should lead to improved accuracy in delay estimation.

## 5.3 Preference Rule #2: Destination-Bound Path Segments

Routing decisions in the Internet are made at every hop based on the destination. As demonstrated by the second example of Figure 2(c), the original traceroute from which a segment is derived helps both in reducing the list of candidates as well as improving the delay estimate.

This is consistent with how packets are routed through a network: as a packet enters an AS, the ingress router looks up the destination address of the packet, determines the egress point based on the destination prefix and routes the packet towards the egress point. When we segment traceroute outputs by the AS and create the path segment repository, we keep this destination information with the segments and use it later in path stitching. Using only those segments derived from traceroutes with the same common destination prefix ("destination-bound segments") makes sense as it is consistent with how packets are routed in the Internet.

However, an open question with this rule is whether it is effective at narrowing down the list of candidates. To answer this question we look at the number of prefixes in the traceroute outputs that contribute to each segment. Figure 4 plots the CDF of the number of prefixes per segment. In the figure, we consider only those $2,672$ path segments from the source ASes of the Ark monitors. This is the worst case for us given that those segments are likely to see the largest number of destination prefixes. Half of the segments have only $1$ prefix, and $80\%$ of segments have fewer than $10$ prefixes. Given that segments have so few prefixes, the destination-bound segment rule is very likely to reduce the number of candidate paths significantly.

## 5.4 Preference Rule #3: Most Recent Path Segment

Even after applying the two preference rules described above, we may be left with more than one candidate path. At this point we need to define a final preference rule. For this purpose we decide to rank the candidates according to the time of the actual measurement (from most to least recent). Indeed, end-to-end routes can change at any moment in the network, thus the most recent segment is likely to represent the end-to-end route most accurately. Note that this rule may still lead to multiple paths given the timestamp resolution of the traceroute dataset (that is in the order of 1s). In such a case we return all as answer to the query.

## 5.5 Summary

In this section we have described three preference rules that narrow down the list of candidate stitched paths. We apply these rules and break ties. The path stitching algorithm first applies the proximity rule to reduce the initial path segments in the source AS. Then, it selects inter- and intra-domain path segments towards the destination that are derived from traceroutes with the same destination prefix. Finally, it applies the proximity rule at the destination AS. As a last step, it ranks all the remaining paths (with the same distance from source and destination address) based on the timestamp of the measurement.

Our algorithm applies the preference rules segment by segment from the source to the destination AS, and is locally greedy. Then how good is our estimate from the best possible answer? To answer this question, we compare the delay estimate our algorithm returns to the best estimate found

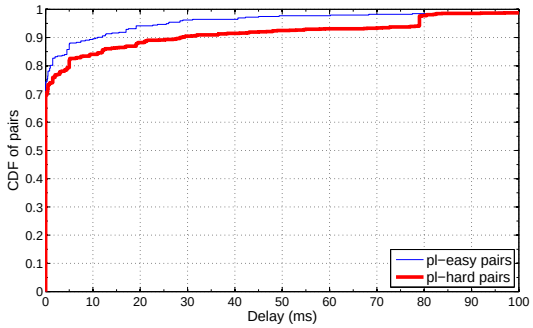



**Figure 5: Difference in delay estimates between most likely stitched and best stitched paths**

with an exhaustive search in the path segment repository. By the best estimate, we mean the closest estimate to the actual measurements. The queries come from our evaluation data sets `pl-easy` and `pl-hard` as described in Section 2.1. Figure 5 plots the CDF of the absolute difference in the two delay estimates. The preference rules perform very well: for 90% of the `pl-easy` pairs and 85% of the `pl-hard` pairs the difference in delay estimates between the stitched path our algorithm returns and the best path that exists in the repository is less than 10ms.

## 6. ADDRESSING SOURCES OF ERROR

In various steps of our path stitching algorithm, errors present in the original measurement datasets enter the final estimates if not filtered. We look into these errors in detail, and explain our approaches in dealing with them. In this section we describe major sources of error and our approach to mitigate their impact.

### 6.1 IP-to-AS Mapping

The first step in the path stitching algorithm maps the IP addresses to AS numbers. Accurate mapping poses two challenges. First, an IP address can be mapped to an incorrect origin AS. Mao *et al.* [24, 25] have pointed out that IP-to-AS mapping techniques based on BGP tables lead to errors due to route aggregation, interface numbering at AS boundaries, and routing anomalies. Second, an IP address can be mapped to multiple ASes, if they have all announced themselves as origin ASes for that address. This problem is aptly referred to as Multiple Origin ASes (MOAS). We look into these two cases in detail, and explain our approaches in dealing with them.

#### *6.1.1 Single origin AS mismatch*

We perform IP-to-AS mapping in two distinct phases of our methodology. First, to build the path segment repository (Section 3.2) we map all the IP addresses in the traceroute outputs to AS numbers and store the entries by the AS number. Second, in **Step 1** of our path-stitching algorithm, we map the queried IP addresses to their AS numbers. If the mapping is incorrect in either of the two phases, the inaccuracy in our estimates could increase.

Mao *et al.* [25] have reported that inaccurate IP-to-AS mappings may cause (1) extra AS hops, (2) missing AS hops, (3) substitute ASes, and (4) two-hop AS loops. Though diverse in patterns, they manifest as one type of error in our methodology: they all produce path segments with wrong AS information.

Out of the four types of IP-to-AS mismatches above, we can identify two-hop AS loops in our Ark data set. The remaining three types are not detectable without accurate AS paths to compare. Performing IP-to-AS mapping on our data set shows that 8.9 % of traceroutes would lead to two-hop AS loops. Resolving these loops is not straightforward. For example, the AS path of $ABA$ should not happen, as no AS path in a BGP table of an AS contains the owner AS itself. Now, are we sure that it is the host in the middle that is mismatched to AS $B$? Or should the first host be mapped to another AS $X$? Or the last host to AS $Y$? Resolving these loops requires additional information, such as BGP paths extracted from routers at the measurement sources [24, 25]. We do not have access to such BGP information for all the sources in our Ark data set.

We have a choice of not using the traceroute outputs with AS loops in our data set. However, the IP addresses are not bogus, and only the mapping is problematic[6] For this reason, we choose to include those path segments from AS loops.

Figure 6(a) shows how we incorporate information from two-hop AS loops. Basically, we add all possible combinations to our database. First, we add a path segment of AS $A$ with both $x_1$ and $x_2$ inside. Then, we also take segments, $:A:$, $A::X$, and $:X:$, from $AXA$. Only the first half or the second half of the loop is ever likely to be used in our path stitching, for no inferred AS path will have a two-hop AS loop. Note that the original IP-to-AS mapping remains consistent and any query with $x_1$ will be mapped to AS $X$.

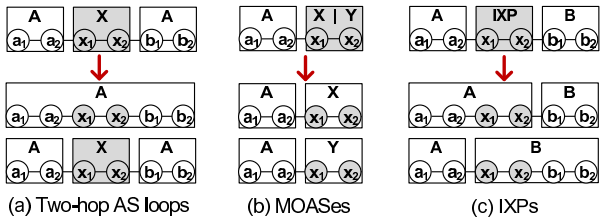



**Figure 6: Three heuristics in IP-to-AS mapping. Circles represent IP addresses and rectangles ASes. $X|Y$ is a MOAS of AS $X$ and $Y$.**

#### *6.1.2 Multiple origin AS*

MOAS conflicts are mostly from multi-homing configurations, Internet exchange point addresses, or anycast ad-

[6]There is a possibility that a route change takes place within a traceroute and an AS loop appears in the traceroute output. We assume it to be a rare event and do not address it in this work.

dresses [26]. MOAS could be also from human errors or malicious activities, such as prefix hijacking. Zhao *et. al.* show that a large fraction of MOAS conflicts appeared only once and did not last more than one day, implying that those short-lived conflicts could be caused by transient BGP misconfigurations [26].

| IP Prefix | MOASes in the same country | % |
|---|---|---|
| 205.189.33.0/24 | 6327 6509 | 26.18 |
| 134.75.20.0/24 | 1237 17579 | 25.41 |
| 207.231.241.0/24 | 293 14221 101 | 3.26 |
| **IP Prefix** | **MOASes caused by IXPs** | **%** |
| 80.81.192.0/23 | 12956 8365 | 10.90 |
| 198.32.176.0/24 | 701 2914 65517 4355 6461 | 7.72 |
| 198.32.160.0/24 | 6461 22691 12989 | 3.58 |
| 206.223.115.0/24 | 293 2914 1273 | 3.02 |
| 195.69.144.0/23 | 286 12956 1200 30132 31283 | 2.78 |
| 206.223.119.0/25 | 2914 293 | 2.33 |
| **IP Prefix** | **Other MOASes** | **%** |
| 69.28.128.0/18 | 22822 21318 | 4.85 |

**Table 4: Prefixes with MOAS conflicts. The percentage refers to the portion of the total number of traceroutes that exhibit a MOAS conflict.**

Let us first analyze the dominant MOAS patterns that we face in our data set. We map all individual IP addresses in traceroute outputs to AS numbers: 472 IP prefixes map to multiple origin ASes. Table 4 lists the top 10 most frequently encountered MOAS prefixes, and these prefixes contribute to 90% of the traceroute outputs with MOAS conflicts. After manual investigation on these 10 prefixes, we find that 54.85% of MOAS cases are located in the same country. We suspect AS 6327 (Shaw Communications) and AS 6509 (CANARIE Network) to be from a customer-provider relation. AS 1237 and AS 17579, MOASes of the prefix 134.75.20.0/24, even have the same AS name, Korea Institute of Science and Technology Information (KISTI). These two ASes are likely to be sibling ASes belonging to the same organization. The third case of 207.231.241.0/24 involves three research networks, AS 293 (Energy Sciences Network), AS 14221 (UW R&D AS), and AS 101 (PNW Gigapop). As we can see there are many different causes of MOAS. However, identifying the actual cause is out of scope for this paper. We are interested only on understanding its impact on the path stitching estimates.

We also observe that 30.33% of MOAS are caused by Internet exchange points (IXPs). Prefixes that belongs to IXPs are usually mapped to MOAS of one or more of participating ASes. IXPs do not appear in BGP routing tables and need a different treatment. To identify prefixes that are assigned to IXPs, we use the information available from the web site of Packet Clearing House (PCH) [27]. One notable example is the prefix 80.81.192.0/23, and its ownership transcends national boundaries of Spain (AS 12956 is Telefonica Backbone AS) and Germany (AS 8365 is Metropolitan Area Network Darmstadt). The `whois` query on this prefix reveals that its network name is DE-CIX-FRA-IXP (Deutscher Commercial Internet Exchange in Frankfurt, Germany). The other five prefixes (198.32.176.0/24 to 206.223.115.0/24) are allocated to PAIX, NYIIX, AMS-IX, EQUINIX-IX-CHI, and EQUINIX-IX-ASH, respectively.

The last prefix, 69.28.128.0/18, belongs to Limelight Networks (AS22822) and Norwegian Open Peering Association (AS 21318). This MOAS conflict is close in nature to the MOAS case from Internet exchange points.

In summary a large fraction of MOAS prefixes are caused by sibling ASes and Internet exchange points in our data set. Still, we do not have solid grounds to give preference to one AS over the others. Therefore, our basic policy is to incorporate all possible combinations of information as in Section 6.1.1. When a prefix maps to MOAS and none of the ASes is an IXP, then we allow an IP address to map to MOAS as in Figure 6(b). If a prefix belongs to an IXP, then we map it to the ASes before and after the IXP as in Figure 6(c) and build path segments accordingly.

The key point in dealing with IP-to-AS mapping is to incorporate connectivity between ASes despite the mapping problem. The above heuristics allow us to include those inter-AS segments in our estimation.

## 6.2 AS Path Inference

Once we map hosts $x$ and $y$ of a query to their origin AS numbers, the next step is to infer an AS path between them. Inferring an AS-level path between two ASes without access to either of the AS is not simple. Much research has focused on AS path and topology inference [8–10,28]. For our work, we use KnownPath [9] as their publicly available tool reports the best accuracy in AS path inference without the first AS hop information [9]. It exploits the AS paths already present in BGP routing tables and infers AS paths by extending these known AS paths. Its inferred AS paths always conform to the valley-free property of AS paths [18]. KnownPath may produce multiple inferred AS paths.

Although KnownPath produces reasonably accurate AS paths, we still see room for improvement in accuracy given our datasets. Mao *et al.* observed that multi-homing is one of the main obstacles to the accurate AS path inference [8], and they provide a novel technique to infer the first AS hop. Qiu and Gao [9] also show that KnownPath's inference accuracy improves by incorporating the first AS hop information. Mao *et al.*'s first AS hop inference, however, requires a designated measurement infrastructure and a large number of active probes, and we cannot adopt their methodology. Our choice is, therefore, to extract first-hop information from the Ark traceroute data. For example, from an AS path $ABCD$ from a traceroute output, we infer that for the destination AS $D$, the first hop ASes of $A$, $B$, $C$ are $B$, $C$, and $D$, respectively. From the Ark data, we garner first hop information for $5,387$ ASes.

It has been reported that links not reported in BGP rout-

ing tables appear in measurements in the data plane [29–32]. ISPs do not advertise all links to their peers via BGP. For example, peering links are not advertised to other peers. Those links are operational, and appear in the Ark traceroute outputs. He *et al.* [31] report 40% more edges and 300% more peer-to-peer edges and Roughan et al. [32] estimate 700 monitors are needed to identify 99.9% edges. As the goal of this work is not to devise a new algorithm to identify missing links, we demonstrate how to incorporate additional information to the existing database of AS topology.

We regard each Ark monitor as a BGP peer, and from each monitor's traceroute outputs, we extract several fields that make up BGP table entries: destination prefixes, next hops (monitor's addresses), and AS paths. We exclude traceroute outputs that do not reach the destination ASes. We also exclude incomplete traceroute outputs (one or more hops are missing). AS paths from traceroute outputs may contain loops or MOASes and we remove those AS paths as well. We investigate in more detail the accuracy of KnownPath for the inferred paths in our data set in Section 7.

## 6.3 Traceroute

### 6.3.1 *Internet dynamics captured by traceroute*

Topology changes and traffic fluctuations are the two main factors that cause network performance change. For this work we use one round of Ark data sets captured in 3 days. For some segments, the delay measurements are already 3 days old by the time we take up the Ark data set. Other path segments are repeatedly updated throughout the 3 days. For example, all traceroutes that originate from the same Ark monitor share the common router hops near that monitor and corresponding segments are updated very frequently.

To calibrate the network performance dynamics in our data, we pick out intra-domain and inter-domain path segments that are updated more than twice and more than ten times. In our data set, 28% of intra-domain and 55% of inter-domain path segments are updated more than twice, and 6% of intra-domain and 22% of inter-domain segments more than ten times. For those path segments, we calculate the standard deviation ($\sigma$) of the delay measurements and plot it in Figure 7. We observe that about 80% of intra-domain and 85% of inter-domain path segments (updated more than 10 times) have $\sigma$ less than 10 ms, while 0.025% of intra-domain and 0.013% of inter-domain segments have $\sigma$ larger than 100 ms.

Given the variability in the data, we need to decide which delay our algorithm should return for a given path. Some application need the most up-to-date information about delay while others [33] are more interested in the trend (for which the median delay would be useful). Our algorithm is agnostic to the choice. We leave the choice to the application and our query results contain both the median and the most recent measurements.

### 6.3.2 *Nondecreasing delay principle*

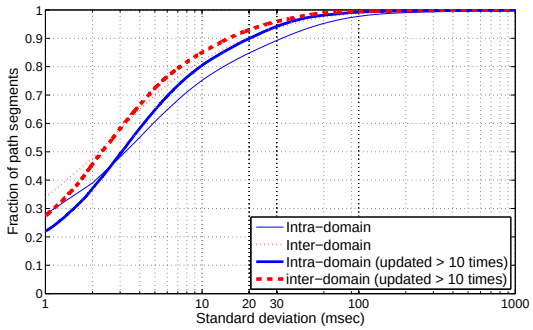



**Figure 7: Standard deviation of delays**

Delay reported at every hop in traceroute should be strictly nondecreasing. However, we observe a large number of traceroute outputs that break this principle. Let us consider the following example:

Traceroute results from *143.248.140.1* to *128.32.255.43*

| | | | |
|---|---|---|---|
| 1 | 143.248.140.1 | 0.236 | *A* |
| 2 | 143.248.117.114 | 0.290 | *A* |
| 3 | 143.248.117.18 | 0.358 | *A* |
| 4 | 143.248.119.2 | 0.487 | *A* |
| 5 | 134.75.20.70 | 0.612 | *B* |
| 6 | 134.75.108.210 | 114.918 | *B* |
| 7 | 207.231.245.129 | 289.926 | *C* |
| 8 | 137.164.27.134 | 133.526 | *D* |
| 9 | 128.32.0.35 | 133.653 | *E* |
| 10 | 128.32.255.43 | 133.401 | *E* |

When building the path segment repository, we break this 10-hop IP path into 7 path segments. During this process, we calculate delay of each segment relative to the segment's first hop latency. For example, delay of the 2-hop path segment (hop 5, hop 6) of :*B*: is 114.306 by subtracting the first hop latency, 0.612. However, the delay for the next segment (hop 6, hop 7) becomes 175.008 and that for (hop 7, hop 8) becomes −156.4. A negative delay of a segment in the middle is very likely an artifact of processing overhead at a router 207.231.245.129.

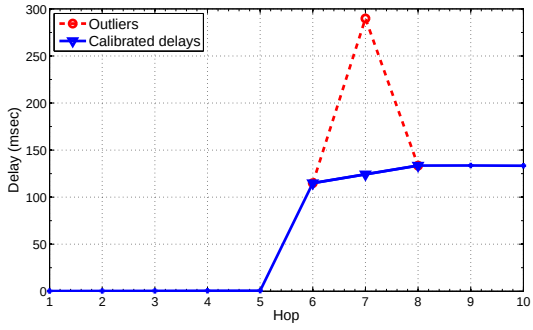



**Figure 8: Filtering outliers in traceroute delays**

Since we use traceroute outputs, we are inherently constrained by the idiosyncrasies that stem from traceroute [34]. One solution could be to seek other data sets that explicitly

measure one-way delays. However, lacking the additional data, we adopt a very simple heuristic to deal with occasional spikes in traceroute-based delay measurements by forcing the measured delays to be strictly nondecreasing. For ease of explanation, we plot the delays from the above traceroute output against the hop number in Figure 8. The dotted line represents the delays from traceroute. If the difference in delay between two adjacent hops (hops 7 and 8 in Figure 8) is below zero, we calculate the differences between the latter hop and preceding hops (between 8 and 6, 8 and 5, and so on), and find the first hop of positive difference (hop 6). Delays from all those hops in between (hop 7) are considered outliers and removed. We then draw a straight line over the outliers (between hops 6 and 8) and extrapolate delays from the line (the solid line in the figure). This simple heuristic has allowed us to retain 60% of Ark data.

## 6.4 Summary

In this section, we have reviewed three sources of error in our methodology: IP-to-AS mapping, AS path inference, and traceroute-based delay measurement. For errors in IP-to-AS mapping, we propose to incorporate all possible combinations so that we do not lose connectivity information between ASes. In order to improve the accuracy in AS path inference, we glean inter-AS segments from traceroute data to exploit knowledge of the first hop AS. For delay measurements that contain segments of negative delay, we propose a simple heuristic to extrapolate reasonable estimates and retain this way more than half of the original Ark data.

# 7. EVALUATION

In this section we demonstrate step-by-step how estimates from our path stitching fare in comparison to on-the-spot actual measurements. For the evaluation, we use the two sets of measurements, `pl-easy` and `pl-hard`, as described in Section 2.1. First, we evaluate the overall quality of AS path inference in **Step 2** of path stitching. Then we show how much value the approximation methods in Section 4 bring to **Step 3**. We employ several rules of preference in **Step 3** when there are multiple candidate segments. We demonstrate how each preference rule reduces deviation in delay estimates from real measurements. As a final part in evaluation, we compare our results against iPlane [1].

## 7.1 AS Path Accuracy

The accuracy of inferred AS paths is critical to our methodology. In Section 6.2 we have proposed to augment KnownPath [9] with AS path information we glean from Ark traceroute outputs. In this section we show the marginal utility of the traceroute data on the quality of AS path inference.

For every pair of source and destination hosts in `pl-easy` and `pl-hard`, we execute **Steps 1** and **2** of our algorithm and obtain *inferred AS paths*. We produce inferred AS path both using vanilla KnownPath (i.e., using only BGP routing tables) and with the first AS hop information derived from the Ark data set. For all pairs in `pl-easy` and `pl-hard` we also have the actual *measured AS paths*.

In order to quantify the accuracy of the inferred AS path we use the Jaccard similarity coefficient as in [35].It considers an AS path as a set of ASes, and calculates the similarity as the ratio of the number of common ASes to the number of ASes in the union of the two AS paths.

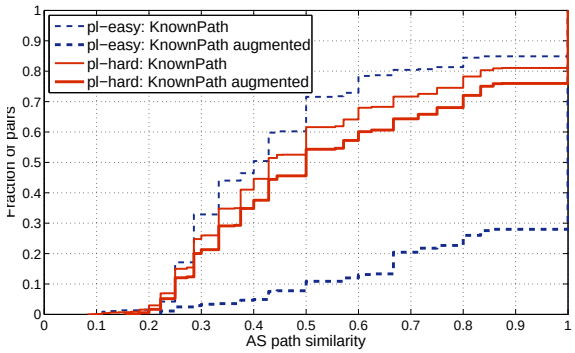



**Figure 9: AS Path similarity**

Figure 9 shows the AS path similarity on the two datasets. The most striking result comes from `pl-easy`. When vanilla KnownPath is used, only 18% of inferred AS paths are exact matches of the measured AS paths. When KnownPath is augmented using traceroute data, the exact matches jump to 72%. This improvement shows the potential value of the additional information. Improvement in AS path similarity for the other set, `pl-hard`, is not as great as in `pl-easy`: only from 19% to 26%. Further investigation reveals that for the host pairs in `pl-hard` there is no first AS hop information from our Ark data set. For the remaining 74% of partially matched paths in `pl-hard`, 77% inferred paths are equal to or shorter than the actual AS paths, and for the 44% shorter paths, 18% of them are a proper subsets of the corresponding actual paths. In the rest of this section all inferred AS paths are from augmented KnownPath.

## 7.2 Approximation Methods

The goal of the approximation methods in Section 4 is to produce *approximate* stitched paths in the face of no stitched path. In this section we show how many more paths the approximation methods allow to find.

For both sets of `pl-easy` and `pl-hard` we stitch path segments along the measured AS paths and inferred AS paths and work with the following four cases: `pl-easy` with the measured AS paths, `pl-easy` with the inferred AS paths, `pl-hard` with the measured AS paths, and `pl-hard` with the inferred AS paths. We compute the fraction of pairs that find stitched paths without any approximation. Then we apply approximation methods one by one (from the most to the least stringent): reverse path segments, router-level and PoP-level clustering, clustering by the /28, /24, and finally /16 prefix. As predicted, the above approximation methods, if applied one by one, show incremental improvement in the

fraction of pairs with stitched paths.

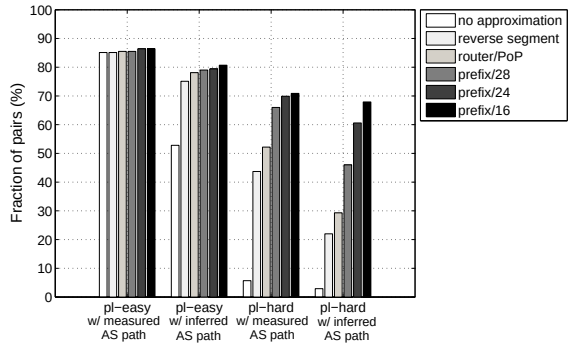



**Figure 10: Fraction of pairs with stitched paths**

The fraction of pairs with stitched paths does not differ much between the measured and inferred AS path cases of `pl-easy`. Because the Ark monitors are co-located with the source hosts of `pl-easy`, 70% of inferred AS paths match the measured AS paths (Figure 9). Yet for 13.6% or 63 pairs, there exists no stitched path. We have two explanations. For 59 or about 12.8% of pairs, their AS paths include ASes or inter-AS links that are not present in the Ark data set. For the remaining 4 or 0.9% of pairs, the path segments cannot be stitched no matter what clustering we use.

The case of `pl-hard` is more complicated. When no approximation is used, only 6% of pairs find stitched paths with measured AS paths and even less with inferred AS paths. Considering the fact that no Ark monitor resides in the same AS as the source hosts in `pl-hard`, even those small numbers are surprising. Those originating ASes have appeared on some routes in the Ark data set. The largest increment in the fraction of pairs comes when we use reverse segments. Further relaxation on clustering constraints show definite incremental improvement. The Ark data set covers about half of the ASes observed in BGP. In order to bring the fraction of pairs with stitched paths from 70% with measured AS paths and 68% with inferred AS paths to the level of `pl-easy`, data sets with a wider coverage of ASes are needed.

Additionally, we repeat the evaluation without router and PoP-level approximations. Resolving router aliases and clustering at the PoP-level require a large number of additional probes. Our interest here is to evaluate our algorithm when those datasets are not available. Indeed, the additional dataset bring limited benefit: clustering with /28 and /24 prefix and without router and PoP, we miss only 5 (0.04%) pairs for the `pl-hard` with measured AS paths case, and 165 (1.6%) pairs for the `pl-hard` with inferred AS paths case. For `pl-easy` cases, there are no missing pairs.

As a last note, clustering by /16 prefix does not bring much gain over clustering by /24 prefix, while the magnitudes of prefixes differ greatly. For the rest of this section, we use clustering by /24 and do not use clustering by /16.

## 7.3 Evaluation of Preference Rules

Approximation methods are useful when no stitched path is found. The opposite case is when there are too many path segments to stitch. In this section we demonstrate how each preference rule reduces the number of candidate path segments, as well as deviation in delay estimates from real measurements.

If an inferred AS path includes a transit AS with a large number of intra-domain segments, it is not practical to consider all possible combinations of path segments. In our path segment repository, the median number of segments of transit ASes in the Ark data set is 25, and the maximum number even reaches 124, 317. If an inferred AS path includes a transit AS with a large number of intra-domain segments, it is not practical to consider all possible combinations of path segments, for we need just one at the end of our estimation process. In this section we demonstrate how each preference rule reduces the number of candidate path segments, as well as deviation in delay estimates from real measurements.

To isolate the effect of preference rules from other factors, we consider only those host pairs in `pl-easy` and `pl-hard` that find stitched paths without any approximation method. We are left with 393 `pl-easy` pairs and 572 `pl-hard` pairs. We do not use the inferred AS path, but use the measured AS paths.

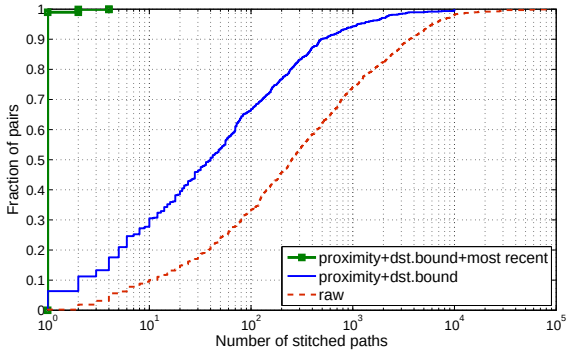
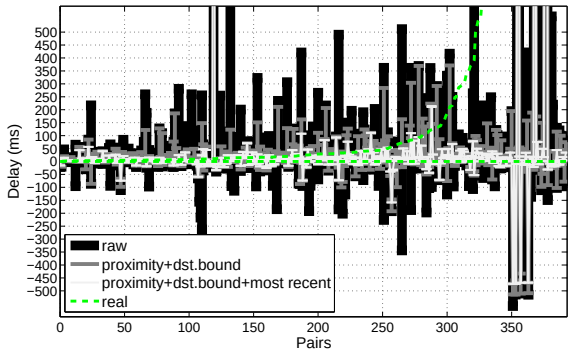


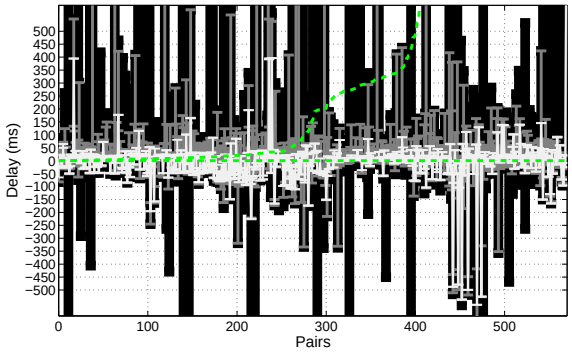

**Figure 11: CDF of no. of stitched paths of `planetlab`**

In Figure 11 we draw the cumulative distribution function of the number of stitched paths per host pair. The dotted red line marked 'raw' represents the total number of stitched paths before we apply any preference rule. We see that almost 60% of host pairs have 500 or more stitched paths. Now we apply the preference rules of proximity and destination-bound path segments and see the number of stitched paths decrease greatly. Still, about 30% of host pairs have about 100 or more stitched paths. Only when we use the segments of most recent measurement, we see the number of stitched paths drop to 1 for almost all pairs. Just a few pairs have more than 1 stitched paths, for traceroute outputs are timestamped with the 1 s granularity and some segments have the same time stamps.

We have shown that the preference rules are effective in reducing the number of stitched paths and thus speeding up




(a) pl-easy




(b) pl-hard

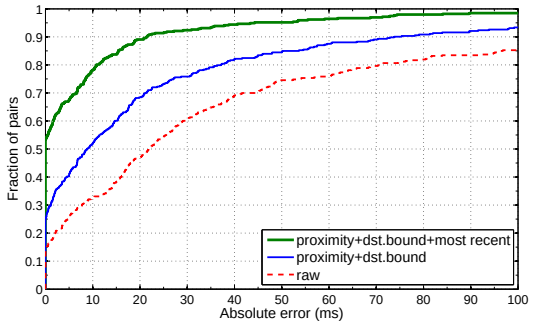
**Figure 12: Improvement in delay estimation**

the estimation process greatly. Now we are interested to see if the reduction in stitched paths has any impact on the quality of delay estimates.

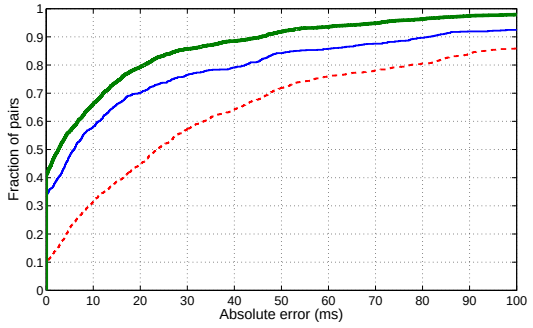
In Figure 12 we plot the minimum and maximum delay estimates of all stitched paths per query against actual measurements. For the ease of illustration, we peg the minimum of measured delays to 0 ms in Figure 12. We draw the difference between the maximum and minimum measured delay as a dashed line. It represents variability in actual measurements. Values that fall between the dashed line and the horizontal line of 0 ms delay are basically indistinguishable from real measurements.

We evaluate the following two combinations of preference rules: *(i)* only with the proximity and destination-bound prefixes and *(ii)* with the first two plus the most recently updated segments. The colors in the bar graph lightens as more preference rules apply. The lightening trend in colors indicates that our preference rules not only reduce the number of stitched paths, but also bring the estimates close to the actual measurements.

For more than 100 pairs in both pl-easy and pl-hard the difference between the minimum and maximum measured delays grows beyond 100 ms. What is the best estimate when actual measurements exhibit such large variability? We leave this choice to the users of our path stitching. We conduct our evaluation with the most recent measurements, but will leave options for users to obtain other metrics about the stitched path when we implement path stitching as a system.




(a) pl-easy



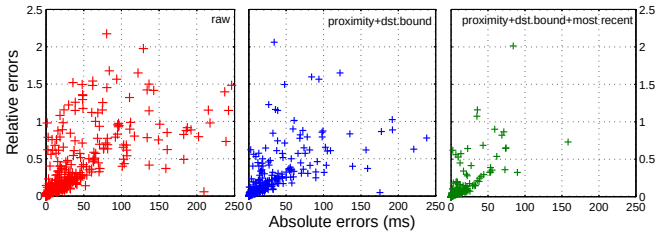

(b) pl-hard

**Figure 13: Improvement in absolute error**

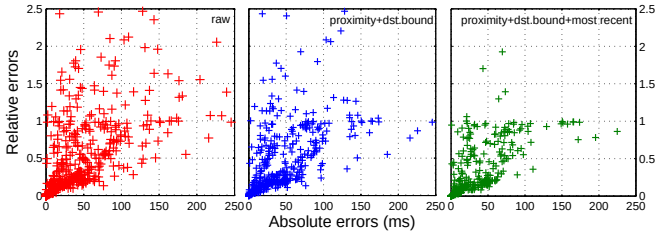
We summarize the trend in Figure 12 by plotting the cumulative distribution of absolute errors in Figure 13. By absolute error we mean the sum of delay estimates that fall outside the minimum and maximum measurements. If both the minimum and maximum estimates both fall within the dashed lines, then we consider the absolute error to be 0 (i.e., the stitched paths are as good as the actual measurements). Figure 13 shows that the absolute error is the smallest when all the three rules of preference are used. Only 10% of pairs have absolute error greater than 20 ms. We observe very similar improvement in delay estimation with preference rules in pl-hard.

Finally, we investigate whether improvements in absolute errors reflect similar improvements in relative errors. We define the relative error as the absolute error divided by the minimum delay measurement (the value that maps to the $x = 0$ for each host pair) and scatter-plot it in Figure 14. From left to right, more preference rules are applied and the spread of relative errors diminishes for both pl-easy and pl-hard.

We conclude that preference rules are effective in reducing the number of stitched paths, and at the same time bring delay estimates close to the actual measurements in both absolute and relative errors.




(a) pl-easy




(b) pl-hard

**Figure 14: Relative error vs. absolute error**

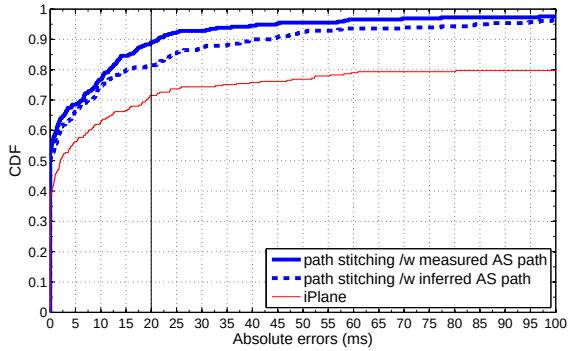
## 7.4 Comparison with iPlane

In this section we compare our methodology against iPlane. Most latency prediction systems based on network coordinates require full-mesh type of measurements between participating nodes and are not amenable to take the Ark data set as input [4].

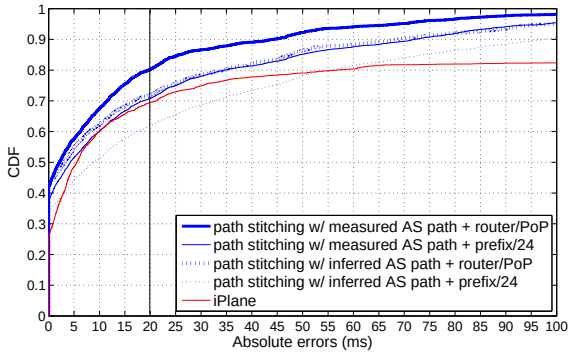
We choose to compare our algorithm to iPlane [1] that uses a similar structural approach to latency prediction and it has been shown to provide more accurate results than other solutions [4, 14]. iPlane reports an absolute latency error of less than 20 ms for 77% of paths. In our case pl-easy shows less than 20 ms for 90% of the pairs (Figure 13), while pl-hard return errors below 20 ms for 80% of pairs.

For more detailed comparison, we use iPlane to process our data set and examine the outcome. We obtained the binary code of the iPlane system and fed our Ark data set to it and in addition related data[7] (PoP and router-level clustering information and IP-to-AS mapping) collected during the same measurement period as the Ark data set. By doing so, we effectively turn the Ark monitors to iPlane vantage points. The current implementation of iPlane has more than 100 vantage points and they together combined probe every BGP atom [36]. We use pl-easy and pl-hard query sets for the evaluation.

First, we examine the number of successful answers. Our numbers are 367 with /24 clustering and inferred AS paths in pl-easy and 6103 with the same combination in pl-hard, while iPlane returns 325 out of 462 in pl-easy and 5109 out of 10,077 in pl-hard. With measured AS paths, we got 399 and 7048, respectively (see Figure 10). We find the numbers comparable and for the rest of the evaluation, we use only those pairs that both iPlane and our path stitching return answers.




(a) pl-easy



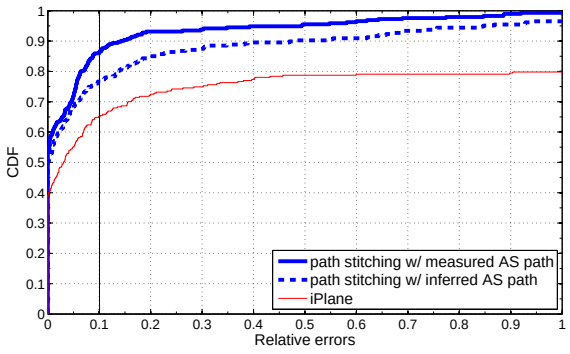

(b) pl-hard

**Figure 15: Absolute errors**

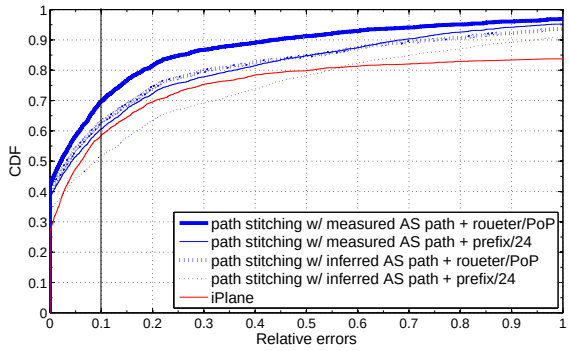
In Figure 15 we plot the CDF of absolute errors. In the case of pl-easy (top graph), we only show results with /24 clustering because other approximation methods return almost the same results. Path stitching reports consistently smaller absolute errors. We note that the iPlane performance observed in Figure 15 is comparable to the best case reported in Figure 4 of [35]. As we have seen in Figure 10 a relatively large fraction of pairs are stitched with approximation methods in the case of pl-hard. Therefore, we draw separate graphs for pairs with different approximation methods. We draw a graph for the results with the router and PoP clustering and with the /24 prefix-level clustering. Overall, path stitching shows consistently better performance with very small absolute errors (below 5 ms.) Path stitching with router and PoP clustering performs very close to iPlane below 35 ms, and gradually shows better performance afterwards. Path stitching with inferred AS path and /24 clustering shows better performance than iPlane only after 50 ms. In both plots of pl-easy and pl-hard the performance of iPlane does not improve much beyond 50 ms. As we do not have access to the source code, we cannot provide further explanation. One conjecture we have is iPlane does not incorporate any data filtering mechanisms, as suggested in [37] or in Section 6.3.2, and anomalously large delays could have

[7] available from http://iplane.cs.washington.edu

an impact on the tail of the distribution.




(a) `pl-easy`




(b) `pl-hard`

**Figure 16: Relative errors**

In Figure 16 we plot the relative errors. In the case of `pl-easy`, 90% of pairs have relative errors smaller than 0.4 in path stitching, while relative errors in iPlane do not change much beyond 1. As in the case of absolute errors, iPlane performs better in `pl-hard` up to relative errors of 0.6 than path stitching with /24 clustering and inferred AS paths, but otherwise path stitching shows better performance. In both Figures 15(c) and (d) relative errors of iPlane do not converge. We proffer the same lack of data filtering mechanisms as an explanation.

In Figure 16 we plot the relative errors. In the case of `pl-easy`, 90% of pairs have relative errors smaller than 0.4 in path stitching, while relative errors in iPlane do not change much beyond 1. As in the case of absolute errors, iPlane performs better in `pl-hard` up to relative errors of 0.6 than path stitching with /24 clustering and inferred AS paths, but otherwise path stitching shows better performance. In both Figures 16(a) and (b) relative errors of iPlane do not converge. We proffer the same lack of data filtering mechanisms as an explanation.

## 8. RELATED WORK

### *Internet-Wide Measurements*

CAIDA's Ark project is the successor of Skitter and maintains a repository of 10 years worth traceroute outputs from tens of sources to every /24 prefix [21]. Its data has been shared widely by the research community [2]. DIMES expands the coverage of Ark-like centralized data collection by endorsing volunteers who contribute traceroute outputs originating from their machines [39]. DIMES compliments Ark with measurements originating from mostly stub networks. However, it has been shown that additional end-to-end path measurements have to be carefully chosen to bring in significant improvements in terms of network coverage [38]. In order to avoid measuring every path, NetQuest suggests a Bayesian experimental design in choosing active measurements for maximum information [39]. Donnet *et al.* proposes a tree-based exploration of the topology in order to reduce measurement traffic [40].

RouteViews [15] and RIPE RIS [16] are the major sources of Internet routing information. They store snapshots and updates of the BGP (Border Gateway Protocol) routing protocol contributed by many ISPs.

All these studies employ various measurement techniques to derive Internet-wide performance characteristics. In our work, we focus on retrieving relevant data from these existing measurements.

### *Network Performance Estimation*

Instead of taking measurements over every path, researchers have looked into estimation methodologies from a limited number of measurements. One approach is to use landmarks for estimating network distances between two arbitrary Internet hosts. IDMaps deploys landmarks (or tracers) that measure distances between themselves [41]. For the distance between two end hosts, IDMaps selects two nearest landmarks to the source and the destination, and returns the sum of the distances between the hosts and their selected landmarks, and between two selected landmarks. DDM is similar to IDMaps but organizes landmarks into hierarchy to locate the nearest landmarks efficiently [42]. King exploits DNS servers as their landmarks [43].

Another approach to Internet distance estimation is network embedding. The main idea of this technique lies in reducing the dimension of collected measurements to a low dimensional space of the Internet hosts; thus the name, "network coordinates [44, 45]. PIC [46], NPS [47], and Vivaldi [4] take one step further and propose a decentralized approach where each participating host measures and shares the information with other hosts (or peers). More recently, Agarwal *et al.* combine network embedding with geolocation, by initializing network coordinates to the actual locations of the nodes [48]. Their approach shows improvement in convergence time and delay prediction error.

The complexity and inaccuracy incurred by network embedding techniques have been analyzed in [7,49] and in part motivated a structural approach to measurements [3, 35]. In particular, Abrahao *et al.* points out a possible reduction in dimensionality when delay within an AS is considered. Our AS-based segmentation and stitching approach aligns well

with this insight [7].

*Network Performance Estimation As a Service*

In today's distributed applications, network performance data sharing among end users is limited. Aggarwal *et al.* have proposed an oracle service hosted by ISPs [50]. This oracle service ranks the queried peers according to certain metrics such as the number of AS hops to the users. ISPs do not need to measure performance, as they already have direct access to customers' bandwidth and link delay information. The oracle service is mostly for peer-to-peer applications and relies on peer-to-peer applications to respect the ranking. Madhyastha *et al.* [1] have end hosts run active measurements and return estimated latency, loss rate, and capacity to their iPlane system in order to construct an annotated atlas of the Internet. Our work is a founding block to a network-wide estimation service similar in spirit to the above two services.

# 9. CONCLUSIONS

In this work, we have presented and evaluated *path stitching*, a new approach to end-to-end Internet performance estimation. Existing measurement systems are naturally limited by the number of available vantage points. Instead of deploying yet another measurement system, we have described how multiple data sets from existing infrastructures can be used together to provide relatively accurate performance measurements of uncharted portions of the Internet. We show that our path stitching approach performs comparably to existing measurement systems that require extensive new data collection campaigns.

Future work will focus on incorporating additional datasets and focus on other metrics of interest to distributed applications such as loss rate, available bandwidth, etc. We also plan to demonstrate the impact of path stitching in the development and performance of common distributed applications. In that context we are currently working on a DNS-like interface that would allow existing distributed application to issue queries about end-to-end path quality and performance. This would allow to better understand the real impact of estimate errors on the end-user experience.

# 10. REFERENCES


[1] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani, "iPlane: An information plane for distributed services," in *USENIX OSDI*, November 2006.

[2] Ming Zhang, Chi Zhang, Vivek Pai, Larry Peterson, and Randy Wang, "Planetseer: Internet path failure monitoring and characterization in wide-area services," in *USENIX OSDI*, August 2004.

[3] Bernard Wong, Aleksandrs Slivkins, and Emin Gun Sirer, "Meridian: A light weight network location service without virtual coordinates," in *ACM SIGCOMM*, August 2005.

[4] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM*, September 2004.

[5] Shansi Ren, Lei Guo, and Xiaodong Zhang, "ASAP: an AS-Aware Peer-relay protocol for high quality VoIP," in *IEEE ICDCS*, July 2006.

[6] F. Thomson Leighton, Ravi Sundaram, Andrian Soviani, Matthew Levine, Andrew Parker, Silvina Hanono-Wachman, and Arthur W. Berger, "Method for predicting file download time from mirrored data centers in a global computer network," May 2001.

[7] Bruno Abrahao and Robert Kleinberg, "On the Internet delay space dimensionality," in *ACM SIGCOMM IMC*, October 2008.

[8] Zhuoqing Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang, "On AS-level path inference," in *ACM SIGMETRICS*, Banff, Canada, June 2005, ACM Press.

[9] Jian Qiu and Lixin Gao, "AS path inference by exploiting known as paths," in *IEEE GLOBECOM*, November 2006.

[10] Wolfgang Muhlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig, "Building an as-topology model that captures route diversity," in *ACM SIGCOMM*, September 2006.

[11] Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, Sujoy Basu, and Sung-Ju Lee, "S3: A scalable sensing service for monitoring large networked systems," in *ACIRI Technical Report*, September 2006.

[12] Young Hyun, Bradley Huffaker, Dan Andersen, Emile Aben, Colleen Shannon, Matthew Luckie, and kc claffy, "The CAIDA IPv4 routed /24 topology dataset - Apr'08," `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`.

[13] Yin Zhang, Vern Paxson, and Scott Shenker, "The stationarity of Internet path properties: routing, loss, and throughput," in *ACIRI Technical Report*, June 2004.

[14] Harsha V. Madhyastha, Ethan Katz-Bassett, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani, "iPlane nano: Path prediction for peer-to-peer applications," in *USENIX NSDI*, April 2009.

[15] Advanced network technology center and University of Oregon, "The RouteViews project," `http://www.routeviews.org`.

[16] "RIPE Routing Information Service," `http://www.ripe.net/ris`.

[17] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias, "The interdomain connectivity of planetlab nodes," in *PAM*, Antibes Juan-les-Pins, France, April 2004.

[18] Lixin Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM TON*, vol. 9, no. 6, pp. 733–745, 2001.

[19] Yuval Shavitt and Eran Shir, "DIMES: Let the Internet measure itself," *SIGCOMM CCR*, vol. 35, no. 5, pp. 71–74, 2005.

[20] Antonio Nucci, Supratik Bhattacharyya, Nina Taft, and Christophe Diot, "IGP link weight assignment for operational tier-1 backbones," *IEEE/ACM TON*, vol. 15, no. 4, 2007.

[21] Brian Fortz and Mikkel Thorup, "Optimizing OSPF/IS-IS weights in a changing World," *IEEE JSAC*, vol. 20, no. 4, pp. 765–767, 2001.

[22] Ramesh Govindan and Hongsuda Tangmunarunkit, "Heuristics for Internet Map Discovery," in *IEEE INFOCOM*, March 2000.

[23] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson, "Measuring ISP topologies with Rocketfuel," in *IEEE/ACM TON*, 2004.

[24] Zhuoqing Morley Mao, David Johnson, Jennifer Rexford, Jia Wang, and Randy Katz, "Scalable and accurate identification of AS-level forwarding paths," in *IEEE INFOCOM*, March 2004.

[25] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy Katz, "Towards an accurate AS-level traceroute tool," in *ACM SIGCOMM*, August 2003.

[26] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang, "An analysis of BGP multiple origin as (MOAS) conflicts," in *ACM SIGCOMM IMW*, SF, USA, November 2001.

[27] "Packet clearing house," `http://www.pch.net`.

[28] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *IEEE INFOCOM*, June 2002.

[29] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger, "Towards capturing representative AS-level Internet topologies," *Computer Networks*, vol. 44, pp. 737–755, 2002.

[30] Rami Cohen and Danny Raz, "The Internet dark matter - on the missing links in the AS connectivity map," in *IEEE INFOCOM*, April 2006.

[31] Yihua He, Georgos Siganos, Michalis Faloutsos, and Srikanth Krishnamurthy, "A systematic framework for unearthing the missing links: measurements and impact," in *USENIX NSDI*, April 2007.

[32] Matthew Roughan, Simon Jonathan Tuke, and Olaf Maennel, "Bigfoot, sasquatch, the yeti and other missing links: What we don't know about the AS graph," in *ACM SIGCOMM IMC*, October 2008.

[33] Youngki Lee, Sharad Agarwal, Chris Butcher, and Jitu Padhye, "Measurement and Estimation of Network QoS among Peer Xbox 360 Game Players," in *PAM*, April 2008.

[34] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao, "A measurement study of Internet delay asymmetry," in *PAM*, April 2008.

[35] Harsha V. Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring, and Arun Venkataramani, "Structural approach to latency prediction," in *ACM SIGCOMM IMC*, October 2006.

[36] A. Broido and kc claffy, "Analysis of routeviews BGP data: policy atoms," in *Network-related Data Management Workshop*, 2001.

[37] Jonathan Ledlie, Paul Gardner, and Margo Seltzer, "Network coordinates in the wild," in *USENIX NSDI*, 2007.

[38] Paul Barford, Azer Bestavros, John Byers, and Mark Crovella, "On the marginal utility of network topology measurements," in *IEEE INFOCOM*, SF, USA, November 2001.

[39] Han Hee Song, Lili Qiu, and Yin Zhang, "Netquest: A flexible framework for large-scale network measurement," in *ACM SIGMETRICS*, June 2006.

[40] Benoit Donnet, Philippe Raoult, and Timur Friedman, "Deployment of an algorithm for large-scale topology discovery," *IEEE JSAC*, vol. 24, no. 12, pp. 2210–2220, 2006.

[41] 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 TON*, vol. 9, no. 5, 2001.

[42] Wolfgang Theilmann and Kurt Rothermel, "Dynamic distance maps of the internet," in *IEEE INFOCOM*, March 2000.

[43] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in *ACM IMW*, November 2002.

[44] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti, "Lighthouses for scalable distributed location," in *IPTPS*, February 2003.

[45] Yuval Shavitt and Tomer Tankel, "Big-bang simulation for embedding network distances in euclidean space," in *IEEE INFOCOM*, March 2003.

[46] M. Costa, Miguel Castro, Antony Rowstron, and Peter Key, "PIC: Practical Internet coordinates for distance estimation," in *ICDS*, March 2004.

[47] T. S. Eugene Ng and Hui Zhang, "A network positioning system for the Internet," in *USENIX Conference*, June 2004.

[48] Sharad Agarwal and Jacob R. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *ACM SIGCOMM*, August 2009.

[49] Eng Keong Lua, Timothy Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft, "On the accuracy of embeddings for Internet coordinate systems," in *ACM SIGCOMM IMC*, October 2005.

[50] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler, "Can ISPs and P2P users cooperate for improved performance?," *SIGCOMM CCR*, vol. 37, no. 3, pp. 31–40, 2007.