Applications of the IP Timestamp Option to Internet Measurement

Justine Sherry

A senior thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Science
With College Honors

Computer Science & Engineering
University of Washington
March 2010
Updated June 2010

Abstract

Limited support and inconsistent behavior for IP options has led to the common belief that most options cannot be useful. IP timestamp was no exception, until recently, when prespecified IP timestamp measurements were successfully integrated into the Reverse traceroute system. Prespecified timestamps allow the sender to request a series of up to four timestamps from all routers which may receive the packet. Each machine that receives a timestamp probe checks to see whether the next, unstamped IP address matches its own, and if so, provides a timestamp.

In this paper, we provide a more extended argument for the role of prespecified timestamps in the measurement toolkit, and demonstrate two new practical use cases for timestamp measurements. We discover over 47% of ICMP ping-responsive routers in our sample support timestamp options, and both document and suggest methods for dealing with some of the differences between implementations of the timestamp option. Where supported, timestamp probes can take advantage of several attributes uncharacteristic of other common measurement tools: a view of the complete path the probe takes, combined responses from up to four requested addresses, and timestamp values from each responsive address.

Illustrating these characteristics, we provide several use cases for timestamp probes. First, we describe how timestamps are integrated into the Reverse traceroute system, discovering routers on the path taken from a destination back to the requesting source. Next, we confirm IP aliases by combining multiple timestamp requests for candidate alias pairs in the same probe. In our application, we identify thousands of alias clusters currently unaddressable by the leading alias resolution technique. As our final use case, we apply the literal timestamp values to assessing one-way link delay. With this technique, we measure the delay between backbone PoPs on the Internet2 network with values comparable to those discovered through measurements at the source.

Presentation of work given on February 23, 2010

Thesis and presentation approved by:

Date:

Contents

1 Introduction

Internet measurement aims to provide descriptive data about the topology, latency, and traffic flows of the global Internet, and make use of this data to better understand how to debug and improve the Internet. The Internet is controlled by many parties and made up of diverse smaller networks. This leads to a situation in which everyone benefits from greater understanding of the network, but no individual group is in control of the entire network and able to make all such information available. As a result, information about very simple properties of far-away connections in the network is useful to many, but difficult to obtain.

Active measurements, sending out specially crafted packets to targeted destinations in order to analyze the responses received, can answer many questions about paths or routers on the Internet, regardless of who controls the network. For example, *traceroute* [1] is a popular tool which reveals the IP-level forward path taken by packets to a distant host, identifying every router along the way from source to destination. However, many important questions remain difficult or unaddressed by current active probing techniques.

Several optional extensions to the IP (Internet Protocol) header (a preamble affixed to every packet that traverses the Internet) are defined with manipulation for measurement and debugging in mind. However, most research has dismissed IP options for presumed lack of support, filtering, and inconsistent implementation [2]. On the other hand, several recent works [3, 4, 5] used the Record Route option, in which each machine forwarding the packet records its address to the packet header (limited to the first 9 hops).

This thesis investigates another such IP option, the Prespecified Timestamp Option. Contrary to previous understanding, we find support of the option to be sufficient for many applications. Furthermore, we demonstrate that timestamps have unique attributes which make them useful towards measuring several properties of the Internet that have remained difficult or impossible with current measurement techniques.

In Section 2, we describe in detail the specification and history of the timestamp option. In Section 3 we address claims of poor support and inconsistent implementation. We discover over 47% of ICMP ping-responsive routers in our sample providing some form of timestamp support. By documenting router-specific implementations of the option, we demonstrate that, rather than inconsistent, responses to timestamp requests are predictable and dependable.

Sections 4 - 6 describe how timestamps can be applied to a variety of practical measurement challenges. In Section 4, we demonstrate one application of timestamp measurements: visibility into the reverse path (from destination to source) taken by a packet. Although traceroute makes identification of routers on the forward path easy, the reverse path has remained an elusive portion of the path traveled by a packet throughout this time. In Section 5, we provide another application of timestamp measurements: IP alias resolution. In this section, we develop a technique for validating when different IP addresses belong to the same machine using timestamp requests that prespecify multiple addresses within the same probe. In Section 6, we provide a third application: measuring one-way link latency between routers. We show how to use timestamp values from probes crossing the link to evaluate the latency of the link. Finally, we conclude in Section 7.

2 Background

2.1 Prespecified Timestamps as Defined

Timestamp requests are an optional extension to the IP header affixed to every packet that traverses the Internet [6]. Three 'flags' specify different behaviors of the Internet Timestamp option; the

focus of this thesis is flag 3, 'prespecified timestamps.' For a packet with the prespecified timestamp option enabled, the sender lists up to four IP addresses in the options header. A pointer is initialized to the index of the first prespecified address.

Each machine that receives and forwards the packet will inspect pointer, and evaluate whether or not its own address is the address targeted by the pointer. If it does indeed own that address, it provides a timestamp consisting of milliseconds since midnight Coordinated Universal Time (UTC), and increments the pointer to the subsequent listed address. Note that a machine may own an address and not provide a timestamp, either because the pointer has been incremented past the listed address (it has received that packet before), or because the pointer has not yet reached its address.

It is permissible to provide a value other than a timestamp in the required format. So long as the most significant bit of the timestamp value is high, anything else may be placed in the other 31 bits.

2.2 Notation

To describe a packet sent with a prespecified timestamp request, we use the following notation:

$$(D|ABCD)$$

Where D is the destination of the request, and A, B, C, and D are the addresses specified (in order) in the timestamp option. We may list less than four addresses after the pipe if less than four addresses were prespecified. In cases where the source address requires description, we include a source S:

$$(S \rightarrow D|ABCD)$$

Finally, when we use source address spoofing, sending a probe from a vantage point V with a different source address S listed in the header as the source, we write:

$$(V/S \rightarrow D|ABCd)$$

In our studies, we always include the timestamp option with an ICMP ping, although any other payload may be used.

2.3 Previous Work with Timestamps

Fonseca et al. [2] investigated timestamps and other IP options with the conclusion that IP options were not useful for measurement. While many of their criticisms of IP options are indeed downsides to using IP options, one aim of this work is to demonstrate that 'workarounds' to the few limitations of timestamp measurements are indeed possible. With these workarounds, in combination with the benefits of timestamp measurements, we argue that timestamp measurements are indeed a practical measurement tool.

With reverse traceroute, we were the first to make major use of prespecified timestamps as part of the reverse traceroute measurement system [5]. Some of this work is elaborated on in section 4.

3 Timestamps as Implemented

The specification for prespecified timestamps [6] leaves some ambiguity about correct implementation of the timestamp option. One important piece of functionality left completely undefined is the appropriate response to successive timestamp requests within the same probe for the same

Classification	Number of IPs	Percent of Total IPs
Unresponsive	83002	31.0%
Extra Stamp	15606	5.8%
Zero Stamps	41422	15.5%
One Stamp	40886	15.3%
Two Stamps	59450	22.2%
Three Stamps	40	0%
Four Stamps	27330	10.2%
Total	267736	100%

Table 1: Responsiveness to timestamp probes for a set of 267,736 public, ICMP-responsive addresses discovered by iPlane on May 10, 2010. Each address D was sent a probes requesting $(D|DXXX)$ and $(D|DDDD)$.

address. Should a destination D respond to $(D|DDDD)$ with one or many timestamps? Another open-ended issue is raised by the possibility of non-standard timestamp values being marked on the probe. What types of values might an administrator wish to provide, if not time?

In this section, we investigate implementations of the timestamp option by routers active on the Internet. We also examine the literal timestamp values and the occurrences of non-standard timestamps (3.4) and document anomalous behaviors (3.2).

3.1 Responses to Direct Probing

To investigate actual implementations of the IP timestamp specification, we probed a set of addresses discovered by iPlane [7] on May 10, 2010. iPlane includes traceroutes for hundreds of vantage points to 140,000 prefixes daily. After filtering targets by probing each address with a simple ICMP echo request to test responsiveness, we were left with 267,736 responsive addresses. We sent each address D a request for its own address, in combination with another address X, belonging to a machine at the University of Washington and known not to be on the path to or from D. We probed each address redundantly from 10 distributed vantage points, with requests for the destination's own address D 0-4 times, followed by a request for X to pad the request up to 4 prespecified addresses $(D|XXXX, D|DXXX, D|DDXX, \ldots)$. We included the X to ensure that extra, invalid stamps were not being introduced into the responses we received.

We received timestamped responses from 47.7% of our targets and unstamped responses from 15.5% of targets. The remainder were unresponsive, either due to the router not accepting timestamp requests, or filtering along all attempted paths to the target. We found a variety of behaviors in the responses from our targets, for which percentages are in Table 1. The first four categories represent a "stamp count." Some would provide up to four timestamps for their own address, while others would provide timestamps only once or twice at maximum. However, there were cases of inconsistency ('Extra Stamp'), in which an address which had provided two stamps previously to a request for multiple stamps might respond later to an identical probe with only one stamp. We found that these 'Extra Stamp' implementations were common to all Linux hosts we tested.

We document additional, anomalous behaviors ('Other Anomalous') in section 3.2.

3.2 Anomalous Behaviors

Over the course of our study, we encountered several anomalous response patterns. While these behaviors were rare, they had the potential to provide misleading or confusing information. We

document some of these responses below.

Overstamping. 5.8% of responsive addresses we encountered in our study provided an 'extra' timestamp, with two timestamps provided for a request ($D|DXXX$), the second being invalid because X was not on the path to or from D. All machines running the Linux operating system that we have tested displayed this bug.

Misaligned Stamps. We received several responses to our direct probes with the prespecified IP addresses modified from the original request, and timestamp values set to 0. Multiple consecutive measurements of the same target revealed that the timestamp values were being inserted into the index where the IP addresses were specified, thus appearing as an incrementing IP address.

Extra Stamps for 0's. The option specification allows for up to four prespecified addresses to be provided. Our timestamp ping tool initially specified four addresses, padding out the extra prespecified indexes with 0's if less than four addresses were desired. Surprisingly, these '0.0.0.0' addresses frequently were returned with timestamps. We speculate that some routers (likely stamping in transit, as successive 0.0.0.0 timestamp values varied) treated 0.0.0.0 as a wild card address. Regardless of the cause, the stamps for 0.0.0.0 IPs have not affected our use of the option.

3.3 Responses to Requests In Transit

A router may provide different timestamp behaviors when it stamps while forwarding a packet destined somewhere else, rather than destined to itself. To investigate, we issued traceroutes to a set of random iPlane destinations, and from those destinations, generated a target list from the intermediary routers we discovered between our vantage point and the destination. We took these 8,278 intermediary targets and requested timestamps for them within a probe sent to the destination along whose traceroute we had seen the target.

In the responses we recovered, we found that 56.8% of routers provided 0 stamps, 23.9% provided four stamps, 17.6% provided one stamp, and 1.6% stamped twice.

Most striking is the shrinking of routers willing to stamp twice. Upon further investigation, the deltas between subsequent stamps where we saw two stamps were extremely high, which indicated that a single router was not stamping twice and then forwarding, but instead, stamping once on the forward and reverse path from the destination of the probe. When re-probing the same set of targets directly, we found that almost all routers which stamped twice stamped either once, or zero times when probed indirectly.

3.4 Timestamp Values

The requested value for a timestamp probe is milliseconds since midnight UTC [6]. We find that most routers appear to attempt to provide this value, providing values within a few minutes of the time determined by our local machines.

When requested for multiple timestamps within the same request, most routers will provide the same value in each provided timestamp. However, we did on occasion (5.3% of stamps in a series of measurements over the DisCarte [4] topology) see a timestamp value increment between stamps, presumably an effect of the clock incrementing while the router was constructing a response packet. These increments were almost always an increment of +1, although we did in negligible cases see increments which were larger.

Many routers will also respond with a 'non-standard' timestamp: one in which the first bit is flipped to 1 and the remaining values are something other than the time. We found, however, that these non-standard timestamps were difficult to predict, and often came subsequent to a standard timestamp provided by the same router. In a study of 16795 responsive targets, 479 ever provided

non-standard timestamps. However, of those 479, all of them provided at least some responses including only standard timestamps. Because non-standard timestamps contain unknown values and are only rarely a problem, we simply discard probes that return with non-standard timestamp values.

4 Reverse traceroute

Traceroute is a tool that allows a sender to measure the forward path their packets take towards a remote destination. Invented in the early 80s, it is likely the most well-known tool for diagnosing problems with and measuring the Internet. However, communication across the network is not a one-way endeavor: packets must travel from the sender to the destination, and from that destination back to the sender. Since forward and reverse paths are often asymmetric, traceroute only reveals half of the path involved.

Reverse traceroute [5] complements normal traceroute, by measuring the reverse path from a remote, uncontrolled destination, back to a controlled source. Reverse traceroute incrementally combines three types of measurement, issued from distributed vantage points (PlanetLab [8]), to discover reverse hops from destination back to source. The first two techniques are record route enabled probes and traceroutes; timestamps complete the suite of tools.

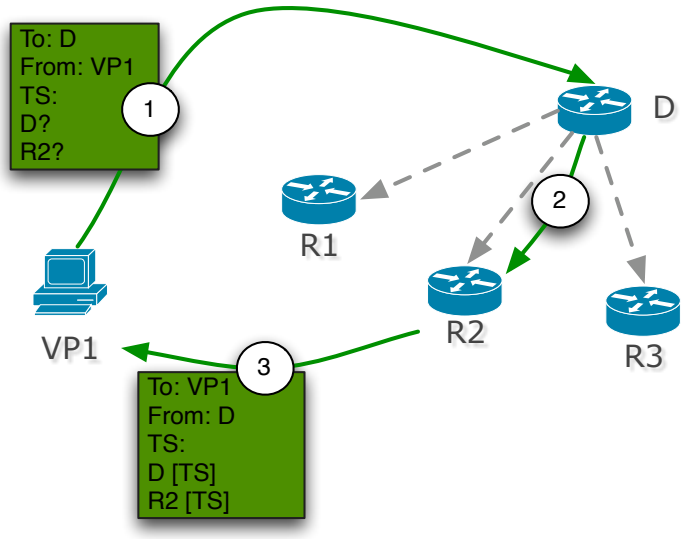


Figure 1: Validating a candidate reverse hop using a prespecified timestamp probe.

4.1 Timestamps within Reverse Traceroute

Timestamp measurements are used to essentially confirm a 'guess' hop along the reverse path. Using recent traceroutes generated by iPlane [7], we generate a set of next hop candidates by aggregating all IPs that we have observed adjacent to R_1 in any previous traceroute (regardless of destination). The next step uses timestamps to decide which next hop candidate is the correct one. For a next hop candidate R_2, we can send a packet $(S \rightarrow R_1|R_1R_2)$. If we receive timestamps for both R_1 and R_2, we conclude that the packet reached R_1, and on its return path crossed R_2, confirming that R_2 is the true next hop. With these measurements, Reverse traceroute takes advantage of the

fact that timestamp requests provide more complete visibility into the reverse path, unlike forward traceroutes which are limited to the forward path, or Record Route probes which are limited to 9 total hops.

However, a working and successful implementation must take into consideration packet filters, timestamp-unresponsive targets, and anomalous behaviors. Below, we describe four modifications to the simple algorithm to limit false positives threatened by Linux hosts, to avoid forward path packet filters, to infer reverse stamps when the candidate reverse hop is also on the forward path, and to infer reverse stamps when the most recent hop does not provide timestamps, but the candidate hop does.

4.2 Modifications to Base Technique

4.2.1 Overstamping

The Linux 'overstamping' bug (in which the machine provides timestamps in the next two open indices if it matches the prespecified address for the first index, but regardless of the prespecified address for the second index) poses a threat of false positives to this technique. Sending a probe to R_1 requesting stamps $\{R_1,R_2\}$ when R_1 overstamps would lead to the inaccurate conclusion that R_2 was on the return path from R_1.

To account for this threat, we first send $(R_1|R_1R_2R_2)$. We have never observed an overstamping machine stamp more than twice, so if the second R_2 request is stamped then it must be that R_2 was actually on the return path. However, many machines provide a maximum of one stamp. Thus, if R_1 is a well-behaving router, and only the first R_2 is stamped but the second is not, distinguishing whether or not R_2 actually stamped remains difficult.

We employ two methods to distinguish whether or not R_2 stamped in this situation. First, we look at the stamp values. While intuitive at first glance, it is not effective to simply require that the stamped values for R_1 and R_2 be unequal. First, we observed 5.4% of Linux overstampers incrementing the timestamp between stamps, bringing a threat of false positives. In addition, in a study of adjacent address pairs within the iPlane topology, we found equal stamps provided by two well-synced neighbors in 6.3% of cases, which would lead to false negatives. Instead, we accept that R_1 and R_2 were stamped by separate routers if the stamps for R_1 and R_2 only if the increment between R_1 and R_2 is large (greater than 3) or if the timestamps decrement.

If the timestamp values leave the stamp unclear, we send a second probe, $(R_1|R_1X)$, where X is a 'dummy' address known not to be on the path. If R_1 stamps into X's timestamp slot, we declare that R_1 is an overstamper. If R_1 does not, we know that the original stamp for R_2 was indeed valid.

4.2.2 Avoiding Forward Path Filters

Some networks filter ICMP Packets, and others may filter options-enabled packets. If a filter lies on the reverse path between the destination and the source, options-enabled ICMP packets will be unsuccessful at discovering reverse hops: the packets will be dropped between the destination and source. However, if a filter lies on a forward path to the destination, but not on the path from destination back to source, it is possible to 'route around' a forward path filter using spoofed probes, and successfully traverse the filter-free reverse path.

Figure 2 demonstrates the efficacy of this technique over a small sample of 1000 random destinations from 662 ASes, all of which were found responsive to timestamp probes in previous experiments. We chose 10 spoofing PlanetLab nodes we found to receive (non-spoofed) timestamp

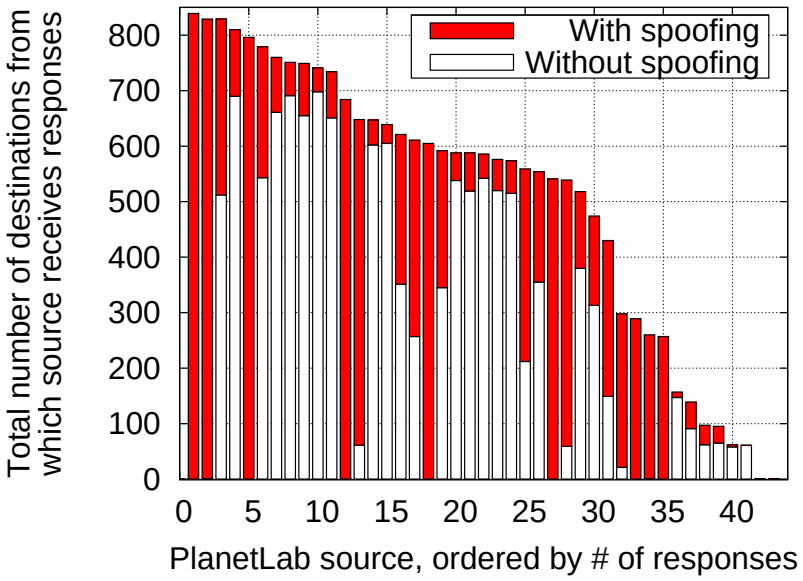


Figure 2: For 43 PlanetLab nodes that receive responses from fewer than 700 out of 1000 random timestamping destinations when sending non-spoofed probes, the number of destinations from which the node receives responses once we include spoofing. The graph shows the total number of unique destinations, first when sending the ping directly, and then when using 10 spoofers. The nodes are ordered by the total number of responding destinations. The PlanetLab sites not included in the graph do not benefit from the technique: 103 do not experience significant filtering, and 63 were either not working properly or are completely stuck behind filters.

responses from the highest number of destinations, and 209 PlanetLab nodes from each up and running PlanetLab site, removing nodes from the same site as our spoofers. First, each non-spoofing node sends a series of timestamp pings to each destination. Then, for each non-spoofing node, each spoofing node sends 10 timestamp pings spoofed as the non-spoofing node, to each destination.

The graph shows responses for a subset of receivers: those which did not start out with high response rates when probed directly (they wouldn't benefit from spoofing because they already receive a high number), and those for which we received no responses at all (either because the site was not working, or the site was completely trapped behind a filter). For each of the 43 remaining receivers, the total number of unique IPs which responded to probes. The top half of the bar includes IPs which were only discovered through spoofed measurements - for many, a significant percentage of the total, if not the entirety.

4.3 Inferring Forward and Reverse Path Stamps

One peculiarity of timestamp implementations is that, although a router may respond to a request for its own IP address with exactly two stamps when probed directly, this is not the case for a router which timestamps while forwarding the packet. As a result, if a response to $(R_1|R_2, R_2)$ is received with two stamps for the same IP address, we can infer that R_2 was in fact a one-stamper

which stamped on both the forward and reverse paths.

4.4 Spoofing to Isolate Reverse Path

When checking if R_2 is on the reverse path from R_1, we normally ask for both R_1's and R_2's timestamp, to force R_2 to only stamp on the reverse path. However, Table 1 reveals 16.6% of routers in the iPlane topology responding to timestamp probes without providing stamps. Hence, we cannot use this technique to always discover a stamping R_2 on the reverse path.

In situations where R_1 does not stamp, we find a vantage point V which we can establish does not have R_2 on its path to R_1. Then, V pings R_1 spoofed as S, asking for R_2's timestamp (but not R_1's). If S receives a stamp for R_2, it proves that R_2 is on the reverse path from R_1.

To test whether R_2 is not on V's path, we send a series of timestamp pings from V to R_1 requesting R_2's timestamp. We repeat the measurement in order to ensure that the packet was not dropped, or that R_2 does not stamp inconsistently. We find that after 10 probes, the chance of R_2 stamping in the next two probes is less than 1%.

4.5 Summary

With Reverse traceroute, we demonstrated two important aspects of working with timestamp probes. First, since timestamp probes can be stamped by a packet in-transit on either the forward or reverse path, timestamp measurements can be used to gain insight into the reverse path taken by packets from a remote host back to the source. Second, we showed that while certain peculiarities due to different implementations or filters can be difficult when working with timestamps, oftentimes they can be resolved with simple workarounds. As a result, timestamps are a practical technique for determine if a particular address lies in the path taken by a probe, even along the reverse path from the destination.

5 Alias Resolution

The mapping of machine to IP address is not always one-to-one: a machine may have more than one IP address. Routers, which serve as hubs for forwarding traffic in many directions, will necessarily have many IP addresses.

Multiple IP addresses belonging to the same machine can create ambiguity with regard to measurement. For example, when two traceroutes are examined, each listing the IP addresses along a single path, one may want to determine whether or not the two paths intersect. This is trivial when a router provides the same IP address to both traceroutes. However, a router may respond with a different IP address to each traceroute, depending on the incoming interface taken by each measurement. Without a way to tell whether or not the IP addresses belong to the same router, it is impossible to tell whether or not the paths intersect. Issues of this sort frequently arise when trying to generate complete topologies of individual networks or compare different paths taken across the Internet.

This problem of recognizing when distinct IP addresses belong to the same machine is referred to as *alias resolution*. While many methodologies have been developed for identifying alias relationships, thus far, no method has been developed which covers resolution of all potential alias pairs. In this context, some have proposed that the future of comprehensive alias resolution will lie in successful combination of techniques with different coverage [9].

This section describes a new technique for alias resolution using prespecified timestamp measurements, which complements existing techniques. In the following subsections, we describe and

evaluate the technique, and then propose how timestamp based aliases fit in to the larger context of alias resolution.

5.1 Applying Timestamps to Alias Resolution

Prespecified Timestamp requests have the uncommon feature of allowing multiple IP addresses to be specifically targeted in a single packet. Our technique nests requests for pairs of IP addresses in a single probe: for a candidate alias pair of IPs A and B, we send four types of requests: $(A|ABAB)$, $(A|BABA)$, $(B|ABAB)$, and $(B|BABA)$. Using timestamp probes in this manner allows us to make two primary observations that lead to the resolution of alias pairs: that certain topological relationships should be impossible between non-alias IP address pairs, and that alias pairs should have a shared clock. Along with a few more limited indicators, these principles can be applied to IP pairs providing two or more timestamps in their responses.

5.1.1 Timestamp Values as a Fingerprint

Literal timestamp values serve as a fingerprint for assessing alias candidacy, by allowing us to identify when two IP addresses appear to share access to a common clock. If we receive multiple timestamps for one of our probes, a single host providing timestamps should provide the same timestamp value for each stamp.

Making use of this, we check the difference between timestamps for all responses to all probes for each pair. However, we have observed rare cases where the timestamp values have some increment between stamps, although the timestamps are provided by the same host (in response to $(A|AAAA)$). Therefore, we allow some leeway. Instead of requiring perfectly matching timestamp values, we say that if less than 90% of the responses have all timestamps equal, or we observe decrements between stamps, we discard the pair.

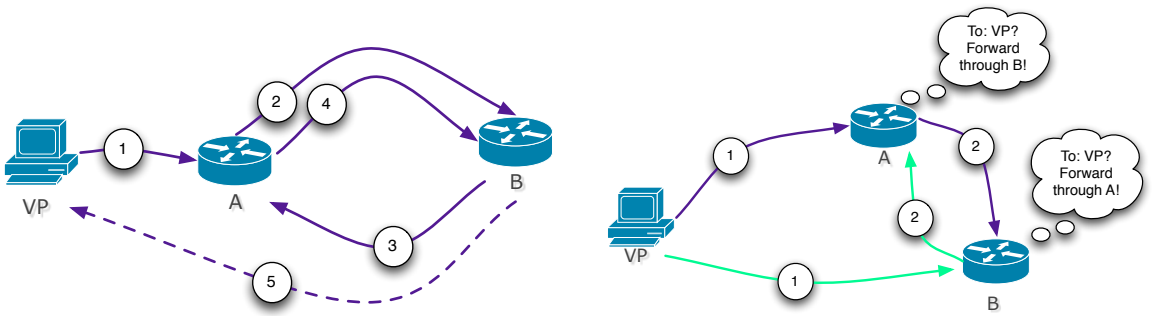
5.1.2 Topological Constraints

Because a prespecified address will only be stamped after all previous addresses have been stamped, the ordering of the timestamps recorded must be the ordering of machines the packet traverses. Using the ordering suggested by combinations of stamps, we can infer constraints on the topological relationship between pairs of IP addresses. We find two cases in which the relationship implies impossible or unlikely conditions if the addresses belong to different machines, and allows us to identify each IP with the same machine.

Loops. Packet forwarding is typically a stateless operation based solely on the destination of the packet. A router, upon receiving a packet, will read the destination of the packet and forward it out on the appropriate interface immediately. Thus, if two routers are configured to forward packets for a certain destination to each other, they should continue forwarding the packet to each other until its time-to-live value expires.

We observe two response patterns with timestamps that imply some sort of looped routing configuration. But, unlike the scenario described previously, the packet obviously managed to escape the loop and return to the destination. (It is possible that the packet escaped the loop because the loop existed only temporarily, as the routers involved updated their routing tables.) That the packet would have to escape such a loop leads us to conclude that the packet did not travel back and forth between two routers, but was at one router and stamped multiple times for multiple interfaces owned by the same router.

There are two separate scenarios in which we infer loops of this sort. The first occurs when four stamps are received in response to any of our queries. Figure 3(a) illustrates the route taken



(a) Configuration of two distinct routers providing consecutive stamps to the request $(A|ABAB)$.

(b) Configuration of two distinct routers providing consecutive stamps to the requests $(VP \rightarrow A|AB)$ and $(VP \rightarrow B|BA)$.

Figure 3: Two configurations implying routing loops. Assuming destination-based routing, loops should be impossible (or temporary, existing only while the routers are in the process of updating their forwarding tables). Consistently receiving responses that imply such configurations leads to the conclusion that the IP pairs must be aliases.

between by a timestamp request $(S \rightarrow B|ABAB)$. Timestamp values for all four prespecified addresses suggests that the packet passed from A, to B, to A, to B again, meaning that B is configured to forward the return traffic en route to S through A, and A is configured to forward the return traffic en route to S through B, resulting in an impossible loop.

The second scenario, demonstrated in Figure 3(b), is the combination of two-stamp responses to $(S \rightarrow B|BA)$ and $(S \rightarrow A|AB)$ received from the same vantage point S. Receiving these responses from the same source means either that B and A are aliases, or that B's return traffic to S is routed through A, and that A's return traffic for S is routed through B. Because this second explanation is once again a looped scenario, we conclude that A and B belong to one machine and are aliases.

Distance. Oftentimes, a single vantage point will receive timestamps for $(A|AB)$ alone, but not $(B|BA)$. In this situation, we still are able to infer alias relationships by combining the constraint that B lie on A's reverse path back to the source with TTL values returned by packets sourced by both A and B.

TTL values are usually initialized to one of several well-known values. By inspecting the TTL value on a packet sent by a sender D from a recipient S, it is simple to generate an estimated reverse path length by subtracting the returned TTL from the nearest initial value. For a candidate pair A, B which provides a response to the query $(A|AB)$ alone, but not $(B|BA)$, we generate estimated reverse path lengths for both A and B using TTLs returned by probes to each of them.

Assuming A and B are separate machines, we know from the returned timestamp probe that B is on the return path from A. Thus, A's reverse path length should be equal to B's reverse path length, plus some difference X representing the number of hops in between A and B along the reverse path. Should A and B have the same reverse path length, then this X value is 0, meaning there are no hops between A and B, and A and B must reside on the same router.

5.1.3 Non-Alias Pairs

We declare a pair to be a definite non-alias under two conditions. First, we declare a pair to be non-aliases if the timestamp values do not appear to share a common clock. We decide that two routers don't share a common clock when the delta between their stamps within the same packet

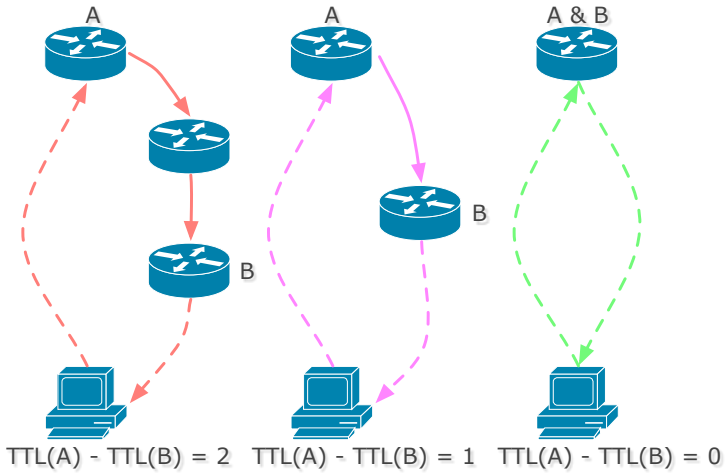


Figure 4: Because TTL values are initialized to a small number of standard values, we can estimate the reverse path length to S from routers A and B. If we know that the reverse path from A traverses B, B's reverse path length to S must be shorter than A's. If their reverse path lengths are the same, A and B must reside on the same router.

is negative, or if it is larger than 1 more than 10% of the time. Second, we declare a pair to be non-aliases if we receive a three-stamp response to one of our probes. Among all of our classified router behaviors, no machines ever provide three stamps to a request.

5.2 Evaluation

To evaluate our technique, we generated a dataset of alias pairs using timestamp measurements. We started with a set of traceroutes taken September 2, 2009 by iPlane [7]. After removing addresses in private prefixes and blacklisted addresses, we had 328,297 distinct IP addresses. Of these, 111,963 were responsive to timestamp requests. Using the TTLs from UDP pings to the responsive addresses, we clustered the addresses into candidate alias pairs using the same technique as described in [7]. After removing any address that over-stamped, removing singleton clusters, and removing clusters in which all addresses provided 0 stamps, we were left with 11,833 non-singleton clusters consisting of 1,848,385 candidate pairs. To each of these pairs we sent four probes: $(A|ABAB), (A|ABAB), (B|ABAB)$, and $(B|BABA)$.

5.2.1 Discussion of Results

For each of the pairs we generated we sent our nested timestamp probes. 1,609,915 pairs responded, 436,627 with no stamps. 1,084,358 provided one stamp, but only 16,285 of those providing one stamp provided a timestamp for a prespecified address different from the destination ($A|B$ rather than $A|A$).

Table 2 displays the categorization of addressable alias pairs: those with two or more stamps. While relatively few candidate pairs from the initial set fall into the 'addressable' category, the vast majority of our candidates are likely non-aliases - recall that they were only declared candidate pairs due to their similar TTL values.

	Fails Clock Test	Passes Clock Test
Four Stamps	37	**16612**
Three Stamps	20732	15
Two Stamps (Loop)	30	**5782**
Two Stamps (Distance)	5669	**9762**
Two Stamps (Other)	4022	*27284*

Table 2: Pair categorization with combined topological and timestamp value conditions. Bolded text (blue cells) are aliases; Normal text (green cells), non-aliases; Italicized text (yellow cells), unknown.

	Four-Stamped	Two-Stamped (Loop)	Two-Stamped (Distance)	Combined
Overlap	8863	3392	4865	15852
True Positive	1547	240	932	2628
False Positive	258	14	150	417
Missed Pair	7058	3138	3783	12807

Table 3: Overlapping pairs with `mrinfo` dataset from same week.

When probed individually (ie. $(A|AAAA)$), targets providing two stamps are three times as likely to appear as targets providing four stamps. However, routers providing four stamps provide more confirmed alias pairs than routers providing only two. In order to confirm a pair which stamps at most two times, multiple probes and measurements must be taken in consideration, while as those with four stamps provide the strongest confirmation of an alias relationship with a single probe and a single condition. Many aliases which provide two stamps may also be counted in the bottom right cell, those which cannot be confirmed as aliases due to a lack of strong topological constraint.

Also of interest is the relationship between the two conditions we apply for alias confirmation. Among pairs which stamp three times, very few at all pass the shared clock test, further strengthening our belief that legitimate alias pairs will not provide three stamps. Among four stampers, and those two stampers we can make a loop argument with, the majority pass the clock test. However, among those two stampers for which we must rely on TTLs and a distance argument, the secondary clock test removes a large subset of pairs. This loss of pairs reflects the relative weakness of the distance argument - it relies on TTLs being consistently initialized and decremented, and probes not encountering load balancing routers. However, bolstered by the shared clock check, we still have reasonable accuracy, as demonstrated in the following section.

5.2.2 Evaluation

We evaluate the accuracy and coverage of our aliasing techniques against a ground-truth dataset of multicast-enabled routers [10].

Testing Accuracy with `mrinfo` Dataset. One property of IPv4 multicast requires multicast-enabled routers to provide a list of its interfaces, associated IP addresses, and neighbors in response to an IGMP `ASK_NEIGHBORS` request. Using the `mrinfo` tool to make daily `ASK_NEIGHBORS` requests, Pansiot et al. [10] mapped remnants of the Mbone multicast overlay network. As a result, they provide daily output of their results covering over four years of measurement. The resulting datasets are limited in several ways: they only include multicast-enabled routers, they are limited to routers which are connected across Mbone to the vantage point used to initiate the requests, and they are

limited to networks which do not filter IGMP traffic. However, because their alias results come from a direct request to the router, they have the exceptional quality of ground-truth accuracy, and complete coverage on all interfaces for each router they address.

Using the results generated with `mrinfo` on September 12, 2009 (the same week our timestamp probes were sent), we compare our timestamp-generated aliases results with the ground truth, when available. The `mrinfo` dataset covers 2869 routers, discovering 123,636 alias pairs over 17,159 IP addresses.

We consider the *overlap* of the two sets to be the set of pairs from either set which contain one or both IPs in both sets. Table 3 provides a breakdown of the overlap, for each alias category as well as for the combined alias results (the transitive closure of all timestamp-inferred alias pairs). The breakdown of the overlap categories are the following:

- True Positive: pairs where both IPs are categorized as aliases with timestamp and by `mrinfo`.
- False Positive: pairs where timestamps categorized an alias, but `mrinfo` either saw the IPs on separate routers, or didn't reveal one of the IPs at all (because `mrinfo` discovers all interfaces on each router it addresses, this implies that the second IP is on a different router).
- Missed Pair: `mrinfo` discovers the pair, and one or both IPs are seen in the timestamp results.

The largest subset is the 'Missed Pair' category. However, this subset is extremely broad: it includes pairs involving IP addresses not discovered by iPlane's traceroutes as well as pairs where one interface is responsive to timestamp but another is not. Turning to the subsets where `mrinfo` measurements were able to determine accuracy or inaccuracy, the results are 86.3% true positive. Inspecting the different alias categories, the four stamped pairs are 85.7% true positive, the two-stamped (loop) pairs are 94.5% true positive, and the two stamped (distance) pairs are 86.1% true positive. Note that these are relatively small subsets, and only include multicast enabled routers, limited to the subset that happened to appear within iPlane's traceroutes. The two-stamped (loop) set are particularly small.

In addition to evaluating the accuracy of our true positive alias pairs, we compared those pairs that we had declared negative against the same dataset. We declare a negative pair when we receive a response from a pair with exactly 3 stamps, or when we see deltas between timestamp values that are frequently large or that decrement. Despite several thousand pairs in the overlap of `mrinfo` with our negative pairs, we did not encounter even a single 'false negative' when compared to `mrinfo`.

Testing Interface Coverage with `mrinfo`. To understand how completely timestamp measurements address all pairs, we probed a set of known alias pairs and tested how well our timestamp technique identified them. Taking the set of alias pairs discovered by [10] using the `mrinfo` tool on December 29, 2009, we generated our $(A|ABAB)$, $(B|ABAB)$, $(B|BABA)$, $(A|BABA)$ requests and probed from 19 PlanetLab nodes known to have high response rates to timestamp probes. The initial dataset included 9,653 IP addresses, joined in 56,288 alias pairs, covering 1,657 routers. Of the initial 56,288 pairs, 13,454 did not respond to probes. 20,477 provided one or zero stamps (useless to our technique). 18,775 provided four stamps, and 202 provided two stamps for both $(A|AB)$ and $(B|BA)$ requests, leading to the two-stamped (loop) configuration. A further 3380 provided two stamps to either $(A|AB)$ or $(B|BA)$, but not both. However, only 1,718 passed our 'same TTL' test, demonstrating that TTLs can vary even for true aliases.

Of all responsive pairs providing two or more stamps, almost all fit our 'shared clock' criteria, but for 13 which displayed a clock decrement between stamps. This left us with a final count of 20,695 alias pairs confirmed by timestamp, the transitive closure of which covered 22,900 pairs.

Overall, we confirmed pairs on 667 routers, or 40.3% of routers we addressed. For 9 routers, the closure of all timestamp confirmed alias pairs was divided into two sets. For the rest, each router corresponded to only one timestamp cluster.

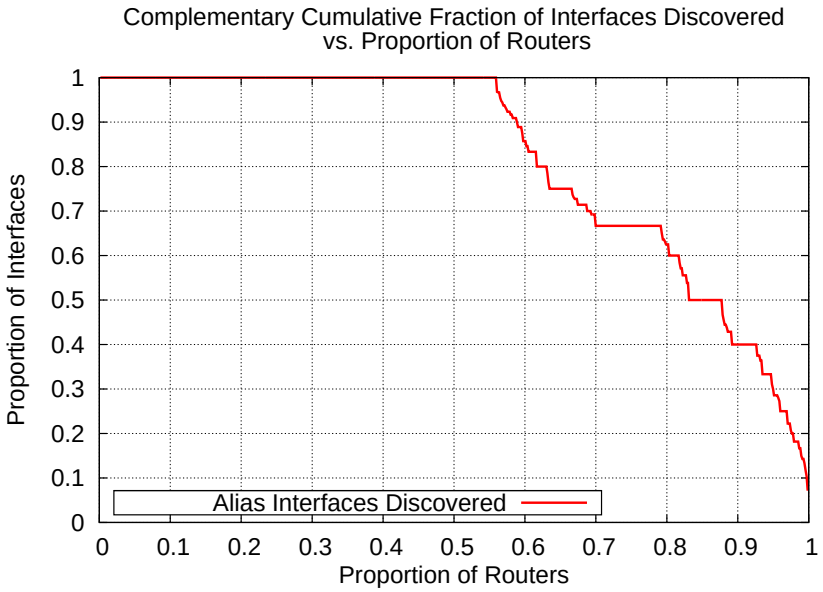


Figure 5: Complementary cumulative distribution of interfaces covered by timestamp measurements over the `mrinfo` dataset. For over half of routers, all interfaces were successfully paired to another interface from the same router.

Figure 5 shows the proportion of interfaces included in the timestamp-generated alias clusters for each corresponding router. All interfaces successfuly identified for over half of those routers which were responsive to any timestamp.

5.3 Alias Resolution in Context

Many techniques exist to address the IP alias resolution problem, but none so far have developed to scale across the entire Internet and all addresses. Mercator sends a UDP probe to a high numbered port, generating an ICMP Port Unreachable error message [11]. In rare cases, the error message will be sourced from an address different from the destination of the original UDP probe, revealing that the two addresses reside on the same router. Ally sent a succession of probes pairwise to suspected aliases and analyzed the IPID value in response packets received [12]. Recognizing that subsequent IPID values are usually on a counter, Ally declares a pair of addresses aliases if subsequent probes interleaved between the two addresses remain in the same range. Radargun [13] borrows Ally's IPID insight and improves on its technique to reduce measurements from $O(n^2)$ probes to $O(n)$ probes. Radargun does this by further noticing that the 'velocity' at which the IPID increments is usually linear. By probing an address several times consecutively, Radargun can measure this velocity and then compare it against the velocity of other addresses, thus probing each address a constant number of times rather than probing all candidates pairwise.

No single technique will suffice to identify all aliases. Looking at existing techniques, Radargun only addresses probe-responsive targets with linear IPIDs, and Mercator, while rarely used, only works in limited circumstances. Thus, some researchers are looking towards effective combinations of existing techniques to provide the type of large-scale alias datasets needed for Internet-scale measurement [9]. As such, we believe that timestamps, while not superseding any individual technique, will play a role in the future of large-scale alias resolution. In one study over a `mrinfo` dataset, we found that 20% of ground-truth aliases that Radargun left 'unknown' were successfully confirmed using timestamp measurements. Therefore, we have reason to believe that timestamps will complement the existing techniques well within the context of large-scale alias resolution efforts.

6 Delay

. Many applications, such as IP geolocation [14], require accurate latency values to function properly. However, traditional efforts have relied on subtracting RTTs from successive traceroute probes, which may be inaccurate due to routing assymetries.

In this section, we describe a technique using the timestamp values in milliseconds to measure the latency of a single link. We demonstrate that despite clock skew and despite a lack of coordination of the target hosts, timestamp probes can measure latency between uncontrolled routers with relatively accurate results.

6.1 Measuring Delay with Timestamps

To measure the latency across a link between two routers A, B, we first issue a probe that crosses the A, B link, and request timestamps from both A and B ($D_1|A, B$)). Using these measurements, we can calculate Δ_1,

$$\Delta_1 = TS_1(B) - TS_1(A) = latency + queue_1 + skew(A, B)$$

In order to isolate the actual link latency, we must first factor out any queuing delay at either router. To do this, we simply issue repeat measurements over a period of time, and take the tenth percentile of several Δ_1's, taking the value for which we assume the queuing component (the only variable component) is minimized. In practice, we take the value at the 10^{th} percentile, in case of anomalous responses.

To factor out skew, we issue a second probe which crosses the link in the reverse direction, ($D_2|B, A$). This resolves to:

$$\Delta_2 = TS_2(A) - TS_2(B) = latency + queue_2 - skew(A, B)$$

Because the probe traverses the link in the opposite direction, the skew this time is opposite what it was in the previous. Once again, we factor out queuing with repeated measurements.

Finally, to calculate the actual latency, we solve for the latency using both of our min Δ's:

$$\frac{\Delta_1 + \Delta_2}{2} = \frac{(latency + skew(A, B)) + (latency - skew(A, B))}{2} = latency$$

Note that identifying paths crossing routers A and B in both forward and reverse directions is not an entirely straightforward task. To identify such a path, we issue traceroutes to routers in A and B's network from hundreds of vantage points, with the intent of discovering as many paths that may cross A or B as possible. Identifying which traceroutes cross the same routers accurately

Cities	TS Latency	OWAMP (Forward)	OWAMP (Reverse)
Atlanta - Chicago	9.5	9.82	9.67
Atlanta - Houston	11.5	11.73	11.79
Atlanta - Washington	6.5	6.78	6.83
Chicago - Kansas	5.0	5.12	5.35
Chicago - New York	13.0	13.25	13.61
Chicago - Washington	18.0	8.35	8.83
Houston - Kansas	6.5	6.98	6.95
Houston - Los Angeles	15.5	17.79	14.13
Kansas - Salt Lake City	12.5	13.28	11.21
Los Angeles - Salt Lake City	11.5	10.69	12.77
Los Angeles - Seattle	12.5	10.62	14.6
New York - Washington	2.0	2.77	2.52
Salt Lake City - Seattle	8.0	7.31	9.29

Table 4: Millisecond latency values for inter-PoP links in the Internet2 network, as estimated by our prespecified timestamp technique, and as provided by OWAMP [15] measurements at the source.

requires successful alias resolution as well, as a single router may respond with a different IP address for different traceroute paths. Furthermore, for every path that we discover, say a traceroute which crosses A, B on the path to D, we in reality issue two probes. We issue the straightforward $(D|A, B)$, that captures A and B on the forward path to D as we have already observed. In addition, we guess that the reverse probe from D may traverse the same path, and send a second probe $(D|D, B, A)$ to catch the reverse link if it is traversed.

6.2 Evaluation

As a trial run for our technique, we measured latencies across the Internet2 backbone. We discovered paths crossing 13 inter-city links, after issuing traceroutes from 482 vantage points, only 121 of which had paths that crossed one of the relevant links and which were able to successfully issue timestamp probes without encountering persistent filters. Table 4 shows our calculated latency values per link. In comparison, we provide latency values provided by OWAMP [15] measurements, which issue one-way pings from one source to another and use synchronized timestamps to calculate one-way latencies. In all cases, except the anomalous Chicago-Washington link, the measurements provided by IP timestamps are visibly close to those provided by the OWAMP project. Excluding the Chicago-Washington link, the average IP timestamp generated value was .46 ms off from the average of the forward-reverse latencies provided by OWAMP. Including the Chicago-Washington link, the offset was only 1.16 ms. These results show that IP timestamps issued from a remote can provide results comparable to those provided by a specialized technique like OWAMP.

6.3 Next Steps

In this section, we demonstrated that IP timestamps can be used to measure link latency without control of either host on the end of the link. We showed that by using multiple vantage points, we could discover traceroute paths that crossed the link in both forward and reverse directions, and that we could use these paths to issue measurements requesting timestamps from hosts on either end of the link.

To apply this technique to more widespread use, practical efforts should attempt to duplicate our

techniques over backbone links of industrial networks. Some challenges for large-scale feasibility will include the limited support for the timestamp option, and a lack of latency data for comparison to validate that the technique is successful on more challenging links. Furthermore, persistent congestion can threaten any technique which relies on probes measuring latency while other traffic is present.

7 Conclusions

We have demonstrated three uses for the IP prespecified timestamp option, all of which solve real measurement problems by exploiting unique characteristics of prespecified timestamps. We used timestamps to discover hops on the reverse path from a destination back to the source, taking advantage of the fact that routers can provide timestamps while the probe is in-transit on either the forward or reverse path. We identified IP aliases by making use of the ability to query multiple IP addresses with a single timestamp probe. We measured the latency of individual backbone links by leveraging the literal timestamp values in milliseconds, provided by routers on either end of the link.

Beyond these use cases, we have have discovered over 25% of active routers providing some support for the IP prespecified timestamp option, making timestamp support prevalent enough for some practical applications. To further future efforts with IP timestamps, we documented each of a limited set of unique router implementations of the option.

Acknowledgements

I would like to thank Ethan Katz-Bassett, Arvind Krishnamurthy, Tom Anderson, Mary Pimenova, Colin Scott, and Harsha Madhyastha for all of their contributions and support.

References

[1] "Tracer T." `http://www.youtube.com/watch?v=SXmv8quf_xM`.

[2] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica, "IP options are not an option," tech. rep., EECS Department, University of California, Berkeley, 2005.

[3] R. Sherwood and N. Spring, "Touring the Internet in a TCP sidecar," in *IMC*, 2008.

[4] R. Sherwood, A. Bender, and N. Spring, "DisCarte: A disjunctive Internet cartographer," in *SIGCOMM*, 2008.

[5] E. Katz-Bassett, H. V. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *NSDI*, 2010.

[6] "RFC 791: Internet Protocol," September 1981.

[7] 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.

[8] "Planetlab." `http://www.planetlab.org`.

[9] K. Keys, "Internet-scale IP alias resolution techniques," in *ACM SIGCOMM Computer Communication Review (CCR)*, 2009.

[10] P. Mrindol, V. V. den Schrieck, B. Donnet, O. Bonaventure, and J.-J. Pansiot, "Quantifying ASes multiconnectivity using multicast information," in *IMC*, 2009.

[11] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery.," in *INFOCOM*, 2000.

[12] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *SIGCOMM*, 2002.

[13] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's growing pains with velocity modeling," in *IMC*, 2008.

[14] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: A comprehensive framework for the geolocalization of Internet hosts," in *NSDI*, 2007.

[15] "RFC 4656: A One-way Active Measurement Protocol (OWAMP)," September 2006.