cing: Measuring Network-Internal Delays using only Existing Infrastructure

Kostas G. Anagnostakis and Michael Greenwald
CIS Department, University of Pennsylvania
200 S. 33rd St., Philadelphia, PA 19104, USA
Email: {anagnost,mbgreen}@dsl.cis.upenn.edu

Raphael S. Ryger
CS Department, Yale University
P.O. Box 208285, New Haven CT, USA
Email: ryger@cs.yale.edu

Abstract— **Several techniques have been proposed for measuring network-internal delays. However, those that rely on router responses have questionable performance, and all proposed alternatives require either new functionality in routers or the existence of a measurement infrastructure. In this paper we revisit the feasibility of measuring network-internal delays using only existing infrastructure, focusing on the use of ICMP Timestamp probes to routers. We present network measurements showing that ICMP Timestamp is widely supported and that TTL-responses often perform poorly, and we analyze the effect of path instability and routing irregularities on the performance and applicability of using ICMP Timestamp. We also confirm that router responses rarely introduce errors in our measurements. Finally, we present a practical algorithm for clock artifact removal that addresses problems with previous methods and has been found to perform well in our setting.**

I. Introduction

Network measurement techniques are crucial for gaining insights into network performance, as well as for understanding the behavior of network control mechanisms such as TCP congestion control [1]. Network parameters (e.g. delay, loss, and throughput) are relatively easy to measure over an end-to-end flow. In contrast, measuring the same quantities on an individual link inside the network is more difficult.

Tools like *pathchar* [2] and similar techniques [3] *directly* estimate internal network performance characteristics such as per-link bandwidth and delays using per-hop round-trip measurements, obtained by eliciting TTL-expired responses from routers. To obtain one estimated sample of the per-link delay, the round-trip time to the head of the link is subtracted from the round-trip time to the tail of the link. Another desirable feature of such tools is that they are able to operate using only the existing infrastructure, without requiring cooperation from remote hosts or routers along some path, and can therefore be used to measure paths to almost any destination. However, there have been several questions about their effectiveness. First, there have been concerns about the representativeness of ICMP responses from routers, due to possible delays in processing such responses in routers. Second, techniques based on TTL-expired responses can only measure round-trip times and not one-way delays. Third, asymmetric paths make it difficult to correlate hop-by-hop round-trip times in order to isolate per-hop estimates. Due to such concerns, direct measurement tools that require no infrastructure support (such as *pathchar*) have been regarded (rightly or not) as unreliable for measuring network-internal delays.

Consequently, several alternatives have been proposed in recent years. Such alternatives are either indirect, or require a measurement infrastructure to be available, or require additional functionality to be added to routers, or some combination of these three. Considerable effort has been put into techniques for inferring internal network performance from end-to-end (*e.g.* active) measurements. These inference techniques, sometimes referred to as *network tomography* techniques, derive network-internal statistics by injecting probe packets from one source to multiple destinations and correlating the observed packet behavior on the resulting tree topology [4], [5], [6].

Other approaches exist, based on passive packet sampling [7], or using the IPMP protocol [8]. While promising, those mechanisms are not likely to be deployed soon on any reasonable scale.

Our work revisits the possibility of using direct measurement, using only existing infrastructure, to estimate per-link network-internal queuing delays. Specifically, we examine the use of ICMP Timestamp probes, and have built a tool, `cing`, which uses them. We have used `cing` to study multiple congestion points in flows over the Internet [9]. We have recently demonstrated [10] that direct measurement techniques can be both more accurate and more robust[1] than inference methods using end-to-end measurements — but only when the direct approach is applicable and practical. This paper answers the question of *when* `cing` is applicable. We explain the technical details needed to make `cing` practical and accurate, detail the problems we encountered, and present the solutions to those problems we solved. Where space permits, we present data that helps explain why our solutions seem most suitable compared to other alternatives that we are aware of. We also measured a large number of paths in order to determine whether the problems that we could not solve limit the applicability of this approach in practice. Some preliminary

[1] After a large number of observations, the mean and median error for direct measurement were moderately lower (more *accurate*) than for indirect measurement. We say `cing` produced more "robust" estimates because the variance in the error was low for `cing` and high for indirect inferencing, and because the mean error for direct methods was low after a much smaller number of observations than for indirect inference.

0-7803-7753-2/03/$17.00 (C) 2003 IEEE

work along these lines was reported in [11].

In summary, the required infrastructure (for example, router support for ICMP Timestamp message) is almost universally deployed, and we have solved a wide enough range of problems, to safely estimate that `cing` is practical and accurate to isolate per-link delays for more than half the paths in the Internet, and per-segment delays for most others. Unlike *pathchar*, we demonstrate that `cing` can accurately isolate one-way delays. Unlike indirect inference techniques, our technique is computationally and conceptually simple and robustly provides accurate estimates. Unlike either, the typical "failure" mode is not an extremely inaccurate estimate, but an a-priori report that `cing` will be unable to provide an answer. The basic limitation of this technique is a susceptibility to the irregularity of IP routing, in particular, the fact that routes from a single source to nodes at both ends of a path segment we wish to measure, do not necessarily follow the same path. We measure how this impacts our ability to perform direct measurements.

The rest of this paper is organized as follows. In Section II we present the measurement technique, as proposed in [11], [10]. In Section III we look into the various problems that can arise with our technique, and quantify their effect with network measurements where necessary. In Section IV we discuss several directions for future work, and in Section V we summarize our findings and present the conclusions of our investigation.

II. Measuring network-internal delays using ICMP Timestamp

A. Preliminaries

We model a network as an arbitrary graph G, with nodes N and links L. Each link, $l \in L$, is an ordered tuple $\langle n_i, n_j \rangle$ in $N \times N$. A link $\langle n_i, n_j \rangle$ implies that nodes n_i and n_j are neighbors, and that n_i can send a packet directly to n_j with no intermediate hops. Let $n_i \rightarrow n_j$ denote a path taken by a packet, in this case from n_i to n_j. $n_i \rightarrow n_j \rightarrow n_k$ denotes a path from n_i to n_k that passes through n_j. When relevant, we denote a direct one-hop path from a node n_i to its neighbor n_j by $n_i \xrightarrow{1} n_j$. We use the notation $\vec{P}$ to represent path variables.

We define the notion of **head**, **tail**, and prefix of a path in the obvious ways. For a node n in a path $\vec{P}$, let n partition $\vec{P}$ as follows. $\vec{P} = \vec{P_1} \xrightarrow{1} n \xrightarrow{1} \vec{P_2}$. Then $\vec{P_1} = \textbf{head}(\vec{P}, n)$ and $\vec{P_2} = \textbf{tail}(\vec{P}, n)$. $\vec{P_1}$ is a *prefix* of $\vec{P_2}$ if there exists some node n in $\vec{P_2}$, s.t. $\vec{P_1} = \textbf{head}(\vec{P_2}, n)$.

Each node n contains a routing map $\mathcal{R}_n(d) : N \longrightarrow N$. $\mathcal{R}_n$ takes a destination node $d \in N$ and returns the next hop $h \in N$. $h = \mathcal{R}_n(d)$ implies that $\langle n, h \rangle$ is in L, and that packets with destination d passing through n travel on the path $n \xrightarrow{1} h$. For all $\langle n, n' \rangle \in L, \mathcal{R}_n(n') = n'$.

Let $s{\rightarrow}d$ represent the path (possibly multi-step) induced by $\mathcal{R}$ on packets travelling from node s to node d. Packets travel on paths based on purely local next-hop routing decisions. That is, $s{\rightarrow}d = s \xrightarrow{1} \mathcal{R}_s(d){\rightarrow}d$.

A map $\mathcal{R}$ is *regular* over a graph G if $\forall n \in s{\rightarrow}d, \forall m \in \textbf{tail}(s{\rightarrow}d, n)$, $\mathcal{R}_n(m) = \mathcal{R}_n(d)$. $\mathcal{R}$ in the Internet is irregular. For example, let $h = \mathcal{R}_n(d)$. $\mathcal{R}_n(\mathcal{R}_h(d))$ does not necessarily equal h. Further, $\mathcal{R}_n$ is not stable in the Internet — routing maps change over time in response to routing updates. The former irregularity significantly impacts which links we can directly measure. In contrast, measurements suggest that, for our purposes, we need not model the latter instability [12]; we verify this assumption in Section III-C.

B. Technique

We can directly estimate the delay on link $l = \langle x, y \rangle$ as follows. Let $s \in N$ denote our measurement source, and t_x and t_y denote the *timestamp difference* returned from ICMP Timestamp Request packets sent back-to-back from s to nodes x and y respectively[2]. The timestamp difference is the difference between the timestamp returned in an ICMP packet and the time the packet was originally sent. If we assume that $s{\rightarrow}x$ is a prefix of $s{\rightarrow}y$, then we can also assume that the two packets will experience approximately the same delay on $s{\rightarrow}x$. In such a case $\delta = t_y - t_x$ will yield an initial estimate of the total delay on l. The total delay includes both the fixed link delay (transmission time and propagation delay over l) *and* the queuing delay on node y.

We cannot assume that the clocks on x and y are synchronized. We must assume that the clocks differ by an offset $O_{x,y}$. (For now, let us assume that the skew in the rate that the clocks advance is small enough so that the change in $O_{x,y}$ over the period of measurement is within error tolerance.) $\delta = t_y - t_x = t_{\text{queuing}} + t_{\text{link}} + O_{x,y}$. Given δ_i and δ_j, two observations of δ, then $\delta_i - \delta_j = \Delta t_{\text{queuing}}$ (because t_{link} and $O_{x,y}$ cancel out).

Let $\delta_{\text{min}} = \min(t_{y_i}) - \min(t_{xi}), i \in [1, m]$ after m observations. Assuming constant clock-offset, for sufficiently large m, δ_{min} denotes with "high" probability the event of zero queuing delay on $x \rightarrow y$. Then $\delta'_j = \delta_j - \delta_{\text{min}}$ gives the desired network-internal queuing delay on path $x \rightarrow y$.

The computation of δ'_j depends upon the path to x being a prefix of the path to y. If we are studying the path $\vec{P} = s{\rightarrow}d$, then we say that nodes $x, y \in \vec{P}$ are members of the same *tomography group*, $TG_{\vec{P}}$, if (i) $s{\rightarrow}x$ is a prefix of $s{\rightarrow}y$, and (ii) $\textbf{tail}(s{\rightarrow}y, x)$ is a prefix of $\textbf{tail}(s{\rightarrow}d, x)$. Tomography groups partition the nodes in $\vec{P}$ into equivalence classes. A node, $n \in P$, is called an *orphan* if $|TG_{\vec{P}}| = 1$.

We can compute δ'_j using direct measurements for any pair of nodes, x, y, in the same tomography group. Condition (i) ensures that the path to x is a prefix of the path to y, and condition (ii) ensures that the segment $x \rightarrow y$ lies entirely in the path $\vec{P}$. We compute tomography groups in three steps. First, we use TTL-limited probes to determine the nodes in $\vec{P}$, by sending packets from s to d and increasing the TTL by 1 until we reach d, as does the `traceroute` tool [13].

[2] Because we require back-to-back packets, we cannot naively use the same mechanism (subtracting the t_x and t_y from the receive time) to measure congestion on the return path. There is no way of generating back-to-back packets from two different destination nodes.

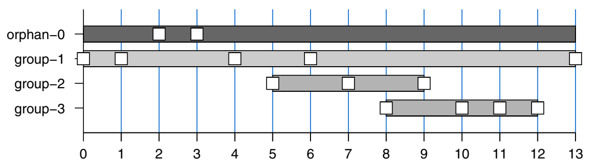

Fig. 1. Orphans and tomography groups on a typical path.

	routers	paths
{TTL, ECHO, TSTAMP}	371800 92.92%	7395 37.22%
{TTL, TSTAMP}	34 0.01%	8 0.04%
{TTL, ECHO}	16853 4.21%	5860 29.49%
{TTL}	5593 1.40%	3537 17.80%
-no reply-	5864 1.47%	3069 15.45%

TABLE I

ICMP TIMESTAMP SUPPORT

The ICMP TTL-expired responses return the list of nodes. Second, we use the TTL technique to compute the path to each node $n \in \vec{P}$. Finally, we examine the path to each node to determine whether each pair of nodes satisfies conditions (i) and (ii). The orphans and tomography groups on a typical path, as determined by this method, are shown in Figure 1.

For simplicity, we assumed above in the computation of queuing delay δ'_j that the clock offset $O_{x,y}$ remains constant. In practice, we cannot make this assumption. In Section III-E we discuss our approach to clock artifact removal.

III. FEASIBILITY STUDY

In this section we investigate the various issues that affect the feasibility and effectiveness of our technique.

A. *Router support for ICMP Timestamp*

We measure how widely ICMP Timestamp is supported in the Internet. Our measurements involve 20,206 paths from the UPENN network to random targets. For each path, we first determine the path using TTL-limited ICMP Echo probes, then, for each node on the path, attempt to elicit ICMP Echo and Timestamp responses. We used a 2-second timeout and 4-retries limit for each probe.

In Table I we present, in the left column, the fraction of routers supporting exactly a given set of ICMP messages, and in the right column, the fraction of paths for which the given set contains exactly the messages supported by *all* routers along the path. We observe that Timestamp is supported on 92.93% of the routers probed. The cases where Timestamp is disabled (while Echo and TTL-expired responses are enabled) is small, at about 4.21%. 1.47% of the routers probed did not respond to TTL-limited probes (while both the preceding and succeeding neighbors on the path responded). In this case we could not determine an IP address to probe for Echo and Timestamp. On 62.78% of the paths there was at least one router that did not support Timestamp. Performing per-link measurements and accurately isolating the sources of delay variation may therefore not always be possible. However, for 37.22% of the paths, all nodes in the path fully support the needed measurement primitives[3]. Note that the paths in the data-sets contain only destinations that reply to ICMP Echo, hence, the figures presented here do not account for networks that block ICMP Echo traffic[4]. On the other hand, the results are conservative due to packet loss and given that we limited the number of retries and timeout for probing each host or router for each of the ICMP-based services.

B. *Accuracy of router probes*

There are several questions with respect to the accuracy of router probes. The first is whether ICMP probes to routers are representative of normal packet behavior. This question arises from the fact that ICMP packets are handled by the slow-path of routers, which are likely to introduce additional delays and therefore bias measurements. Although this possibility has been a concern for router-based measurements, recent preliminary measurements indicate that routers rarely introduce noticeable delays to TTL-limited probes[14]. We conducted two experiments to look further into this matter. We first apply the method of [14] to the case of ICMP Timestamps to confirm that Timestamps behave similarly to TTL-limited probes. The method involves sending pairs of probes, one to the target host, and one to the router under consideration. The source address of the packet sent to the router is spoofed so that it will be deflected to the target host. Because one packet travels directly to the target, while the other one is processed by the router before being forwarded to the target, the difference of the two packets' transit times provides an estimate of router ICMP processing delays. It is important to note that this experiment depends upon the assumption that packet pairs experience the same delay over their common path. In this case, the assumption seems justifiable. First, Figure 5 shows that such an assumption is usually correct. Second, if the assumption is false, and the packets become separated (that is, the second packet is delayed more than the first), then this failure is *conservative*: it will overestimate the apparent ICMP processing time.

As the test requires instrumenting both source and target hosts, a large-scale study does not seem feasible without an infrastructure of reasonable scale. Therefore, the results of both [14] and our own study may not be conclusive. Because of irregular routing, our measurements are restricted to routers that are on the same tomography group as the target host. Our measurements involve 7 sites, with only one that can

[3] Not reflected in Table I are cases of buggy ICMP Timestamp implementations. For instance, some hosts return Timestamps in the wrong byte order. Other problems are detectable, but are not as readily corrected.

[4] A preliminary attempt at quantifying the fraction of hosts that have ICMP probes filtered (either locally or somewhere along the path) using hosts that participate in the Gnutella network indicates that 15051 out of 33533 (44.88%) of hosts reply to ICMP Echo messages. The population of hosts in this experiment may not be representative.

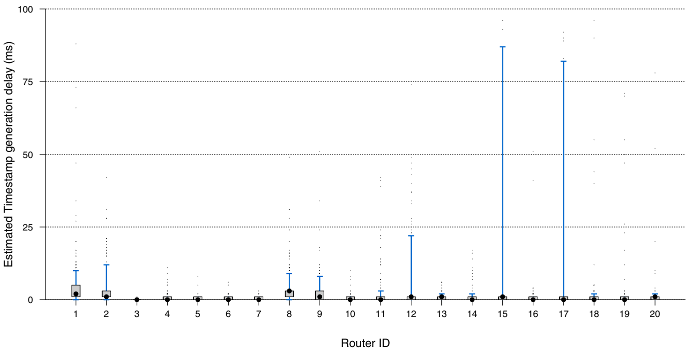

Fig. 2. Distribution of estimated ICMP Timestamp processing delays. Large dots are medians, (barely visible) shaded boxes are inter-quartile ranges, outer bars are 5-95% percentiles, small dots are outliers.

get spoofed packets on the Internet, and 20 routers that are members of suitable tomography groups. The distributions of estimated ICMP Timestamp generation delays are shown in Figure 2. As reported in [14], the median estimated ICMP processing delays are negligible. However, the tails of the distributions are not always well behaved: two out of 20 routers we studied had high 95-percentiles of roughly 80 ms, while another five had non-negligible 95-percentiles between 8 and 20 ms.

We attempt to estimate more accurately how many routers exhibit such behavior, based on the following observation: if for a given link $\langle x, y \rangle$, router x exhibits this behavior, then a fraction of our method estimates will be negative (assuming there is no strong correlation between ICMP generation delays on x and y, in the case that y also behaves similarly). We characterize a router as suspect to this behavior if 10% of the delay estimates are below a threshold of -10 ms. We applied this estimator by taking 100 sample estimates on each of 1368 routers and found that 136 (9.9 %) appear to have this problem.

A more detailed assessment of exactly when and how often this occurs for each router would be useful. Nevertheless, these results indicate that ICMP generation delays will occasionally introduce errors, especially when analyzing tails of the estimated queuing delay distributions. On the other hand, in many cases this problem is detectable, using tests such as the one employed above.

To understand the relative benefits of using Timestamps instead of a TTL-based method, we have obtained measurements on a large number of paths from the UPENN network. We distinguish two cases: measurement of isolated links and measurement of multi-hop segments. For each path, we first determine the irregularity structure and randomly select a pair of suitable measurement nodes. We collected data on 10,931 isolated links and 9,591 multi-hop segments. After applying the clock correction algorithm detailed in III-E, we compare the Timestamp one-way queuing delay estimates with round-trip-based queuing delay estimates. We do the round-trip-based calculation just as it would be done against TTL (Time Exceeded) round-trip data, but we actually do it against the round-trip values for the same Timestamp data used in the one-way estimates, for a direct estimate-to-estimate comparison. The results are shown in Figure 3 for isolated links, and in Figure 4 for the multi-hop segments.

In both cases, there are correlated negative estimates, which can be attributed to occasional ICMP generation delays on the router on the head of the link or segment. Negative round-trip estimates with negligible one-way estimates occur when paths are asymmetric, with the return-path from the head link having larger queuing than the return-path from the tail link. Positive round-trip estimates when one-way estimates are negligible can be attributed to congestion on the measured link or segment if the return path is symmetric, or when the path is asymmetric and the return-path from the tail link having larger queuing effects than the return-path from the head link. Positive one-way estimates when round-trip estimates are negligible are quite interesting and are primarily attributable to uncorrectable clocks we have seen, which flail over as much as tens of milliseconds, as evidenced by fluctuation of apparent one-way times while round-trip times are strikingly stable (negligible or not). The clock correction phase flags such cases, but they are retained in this data set.

Naturally, these effects are more pronounced in the multi-hop segments, as the chance of encountering a congested link increases. An unusual phenomenon in Figure 4 is that round-trip estimates often appear to be within a range between 0 ms and 150 ms higher than the corresponding one-way estimate, with the difference appearing to be uniformly distributed. This appears anomalous to us, and we have several conjectured explanations, but the scale of the experiment does not provide sufficient information to reach any safe conclusions.

To summarize, the comparison of TTL vs. Timestamp-based estimates demonstrated several possible failure modes of TTL-based estimates, and revealed some problems that are common to both approaches. To a lesser extent, several cases were discussed where the clock correction phase allows error through. These errors only affect the Timestamp approach.

Finally, we need to determine how the packet pair assumption affects accuracy. `cing` expects that back-to-back Timestamp probes encounter the same environment up to the head of the link examined. We determine whether this assumption is violated frequently enough to have any noticeable effect on our internal network delay estimates.

We consider again paths to random targets and obtain data on 7,300 routers on those paths by sending 400 pairs of back-to-back ICMP Timestamp probes to each router. We then compute the difference in the Timestamps for each pair, after removing possible errors due to clock resets or adjustments using the clock artifact removal algorithm.

In Figure 5 we present the fraction of routers against the maximum, 98-,95- and 90-percentiles of the computed packet pair error over the run of each experiment. We observe that packet pair error is small in our measurements, meaning that `cing` would provide highly accurate estimates of network-internal delays (assuming all other issues have been addressed). This result should, of course, not be regarded as

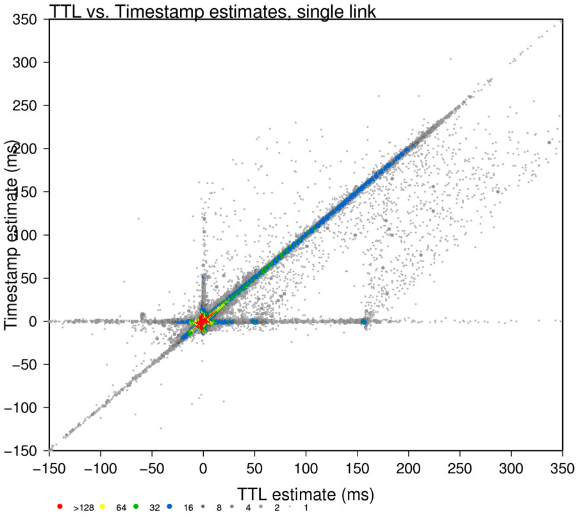

Fig. 3. TTL vs Timestamp estimates, single-hop measurements

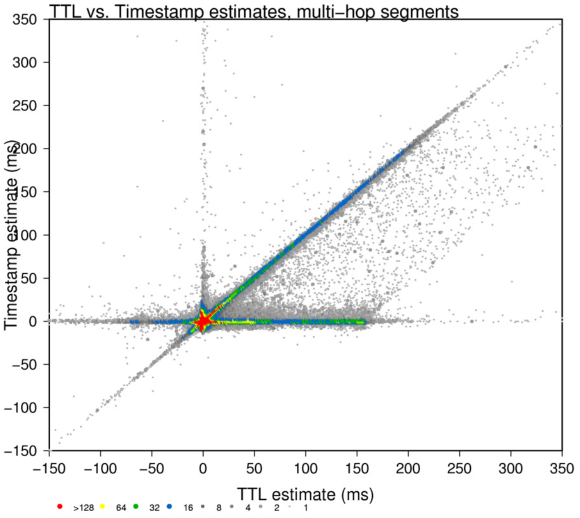

Fig. 4. TTL vs Timestamp estimates, multi-hop measurements

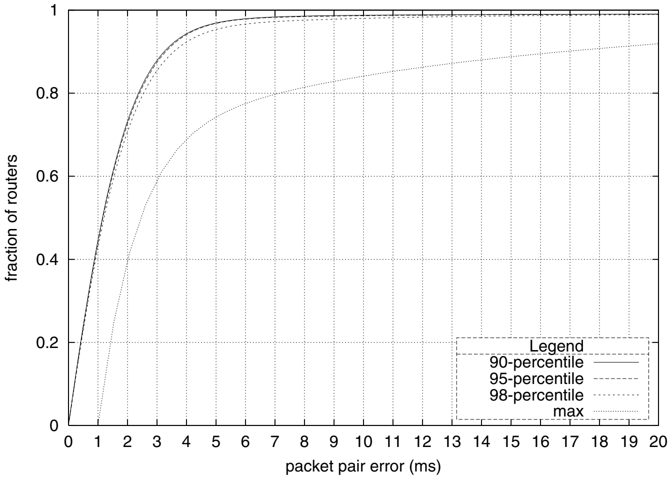

Fig. 5. Measured packet pair error on a collection of routers.

a global property of Internet paths: because of relatively good connectivity of our measurement sources our measurements do not provide insights into situations of heavier congestion or in cases where slow links open up gaps in the packet pair, especially if such phenomena occur near the measurement source.

C. Internal path stability

A rather uncommon property of our technique is that it sends packets not only on the end-to-end path but also on paths to some network-internal nodes (which are frequently not prefixes of the end-to-end path, as we will discuss in Section III-D). The robustness of our technique therefore depends on *the probability that all paths to internal nodes involved in measurement are stable* and not just the end-to-end path, making our technique more sensitive to path instability. While it is known [12] that around 91% of end-to-end paths are likely to be stable over periods of hours, this result does not necessarily account for paths towards network-internal nodes. Although we are not aware of any obvious reason why the same should not apply to network-internal nodes (and one could assume that instability of network-internal paths is not independent), we have performed a limited measurement study to validate this assumption. Our measurements involve 40 repeated path structure samples on each of 1263 random paths measured from the UPENN network within a measurement period of approximately 15 minutes for each path. The results are summarized graphically in Figure 6.

We found that 77.9% of end-to-end paths are stable during the experiment, but closer inspection revealed that 56.9% of the unstable paths were unstable only within the UPENN network (hops 2, 3 and 4) indicating that local multi-path routing within the UPENN network was the primary cause for the observed difference. Discounting for local instability, 90.7% of the paths can be characterized as stable. Analysis of paths to internal nodes on stable end-to-end paths shows further instability in the path structure: only 42.6% had stable paths to each hop. From those paths, 44.7% were found to be locally unstable within the UPENN network and 22.8% locally unstable in other parts of the network. Discounting for local instability, 81.4% of internal paths appear to be stable. The fraction of paths in which both the end-to-end path as well as *all* internal paths are stable is around 76.1%.

These results indicate that it may be necessary to periodically check for possible routing changes, if a significant number of internal nodes are used for measurement. On the other hand, it appears unlikely that many or all internal nodes will be used at the same time. With this condition, we can conclude that although routing instability is somewhat worse for our technique compared to others, it does not significantly affect applicability.

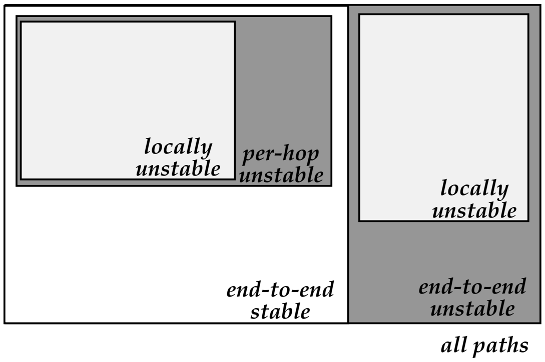

Fig. 6. Graphical representation of path stability results.

dataset	Source	Paths
UPENN	upenn.edu	11,749
YALE	yale.edu	11,742
SRI	sri.com	7,076
UCH	uch.gr	8,553
LIACS	liacs.nl	11,754

TABLE II
DATA-SETS USED IN PATH STUDY

D. Effect of irregular routing

We analyze the effect of irregular routing on our technique, given the constraints discussed in Section II. Most of the results are based on five data-sets containing measurements of characteristics (see Table II) of paths to random target IP addresses[5]. In most cases we only present results for the UPENN data-set; results for the other data-sets display similar properties and are presented in an extended version of this paper[15].

A key metric for the “coverage” of a path in terms of potential measurement nodes is the number of non-orphan nodes. In Figure 13 we illustrate the percentage of non-orphan nodes at a given normalized distance from the source node. We observe that the chance of getting a usable node decreases significantly as we move deeper into the path and closer to the destination. Our technique is therefore more likely to be useful and accurate on links and segments closer to the measurement source. In the proximity of the destination, nodes again tend to be non-orphans with higher probability than in the core. While the general trend is the same for all data-sets, there are notable differences in the magnitude of the irregularities. It is possible to overcome a fraction of these irregularities: in Figure 14 we present the coverage obtained for the paths in the PENN dataset when the PENN node is assisted by the five other nodes in overcoming irregularities.

Non-orphan node endpoints are not sufficient to allow us to measure a particular link. Additionally, we require both nodes to belong to the same tomography group. Thus the fraction of non-orphan nodes tell us which nodes can be used to terminate a measurable segment of any number of hops, but does not tell us whether individual links are directly measurable or not. Figure 15 presents the fraction of individual links per path that are directly measurable, and compares that fraction to the (larger) number of non-orphan nodes. For many paths this fraction is lower than we would like. There are other ways of overcoming routing irregularities, and measuring delay on links that border orphan nodes; cing does not, in general, currently attempt these. For example, cing as described in Section II does not consider combining measurements in the case of overlapping segments (possible extensions to deal with this aspect are discussed in Section IV, and in [16], which discusses an extended version of cing).

There are a number of other interesting characteristics of routing irregularities. Most paths generally have between 1 and 4 tomography groups (Figure 7). The distribution of tomography group diameters, e.g. the distance between the first and last node (excluding those groups including source and destination, whose diameter is equal to path length) is bimodal. The diameter tends to be either relatively small or relatively large, as shown in Figure 8. This also indicates the existence of segments that can be analyzed in detail down to the level of 2 or 3 hops, but also that overlapping of groups e.g. very small groups within larger groups are likely to be common.

We also analyze the relationship between number of tomography groups, path length and number of non-orphan nodes. The ratio of non-orphan nodes decreases with path length, with a significant drop between small (possibly local) paths and medium-length paths, but with slower than linear decrease as we move into longer paths (Figure 12). The fraction of non-orphan nodes per path increases with the number of tomography groups available on a path, but the improvement diminishes as the number of groups per path increases (Figure 10). The number of groups formed by a path is likely to be higher in longer paths, as illustrated in Figure 11.

To better understand the factors behind routing irregularities, we examine how they relate to intra- and inter-domain routing choices. Intra-domain routing is expected to contribute to irregularities, especially considering the existence of parallel paths and adaptive routing such as [17]. Similar effects could also arise at the inter-domain level: network-internal nodes and the target are not necessarily part of the same Autonomous System (AS), and are in principle handled separately by BGP-based routing [18]. To gain insight into this matter, we look at the AS-path structure, examining whether internal paths tend to include ASes that are not on the original end-to-end path. In Figure 9 we show the number of ASes crossed by internal paths that do not appear on end-to-end paths. We observe that for nearly half the paths, all internal paths remain within the end-to-end AS-path, with the number of additional ASes on internal paths rarely exceeding one or two. However, the percentage of paths where paths to internal network nodes include other ASes is significant. From these results, it appears that both intra- and inter-domain routing are responsible for irregular routing and the resulting limitations of cing.

[5] Similar results are obtained when considering targets obtained from Web server traces rather than random targets.

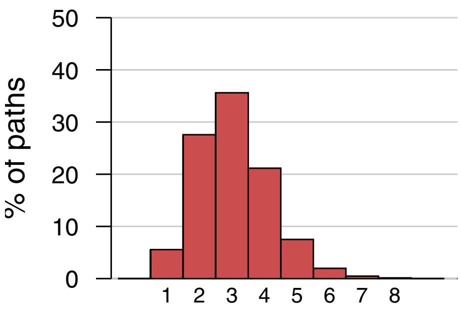

Fig. 7. Tomography groups per path

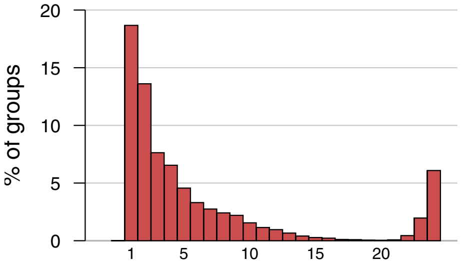

Fig. 8. Tomography group diameter

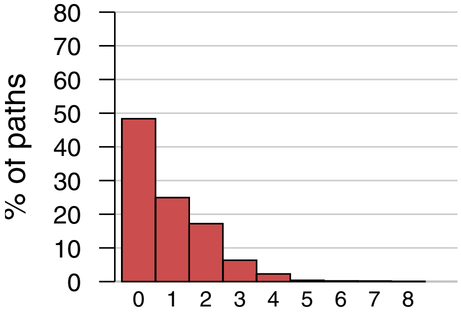

Fig. 9. ASes in internal paths

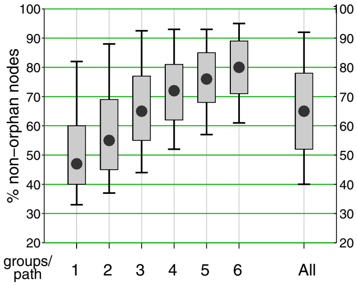

Fig. 10. Groups/path vs. non-orphans

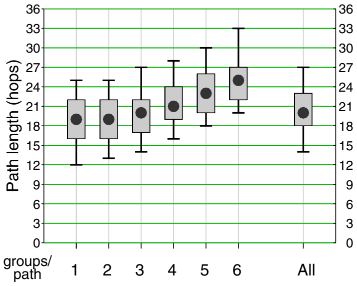

Fig. 11. Groups/path vs. path length

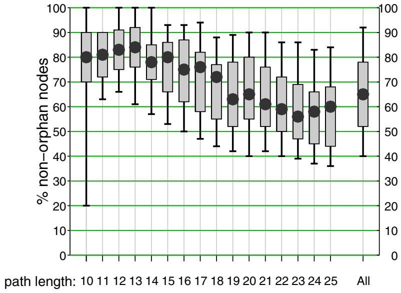

Fig. 12. Path length vs. non-orphans

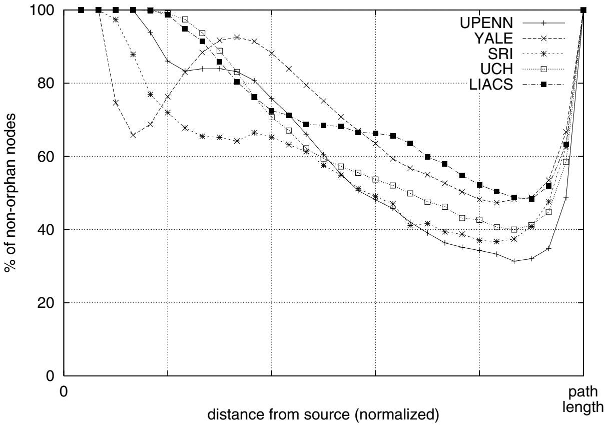

Fig. 13. Percentage of non-orphan nodes vs. distance from source

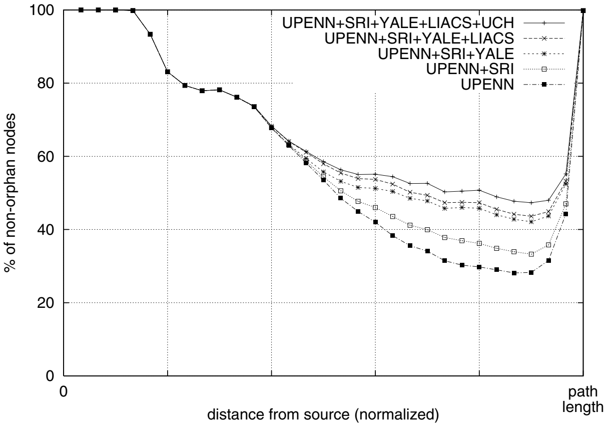

Fig. 14. Percentage of non-orphan nodes vs. distance from source, when multiple sources are used

E. Clock artifact removal: fixclock

Our experiments gather timestamp data from tens of thousands of target clocks, exhibiting a broad variety of behavior. Some clocks exhibit jump adjustments (or “resets”) many times per minute. Some change rate (”skew”) in peculiar patterns. Some report times in milliseconds, but actually appear to have much coarser resolution. In order to reason meaningfully about the collected data, we must correct for such clock behavior if we can, and we must be prepared to drop data that we cannot correct confidently.

Figures 16 and 17 are “before and after” views of timestamp correction. We know the clock on the probing source to be adjustment-free during the probe, so the round-trip times, shown for reference in both graphs, are real. However, the one-way transit times (OTTs) involve the target clock, and

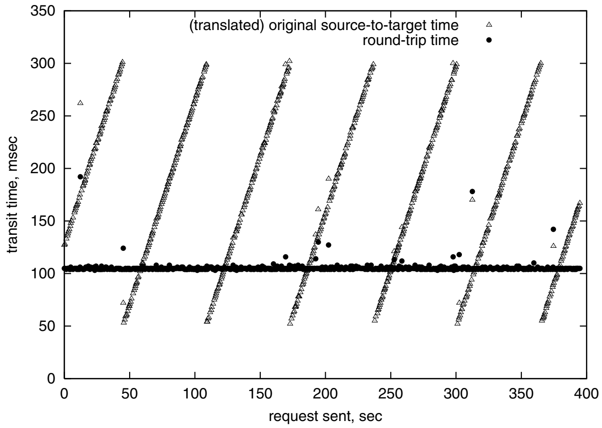

Fig. 16. Before `fixclock` correction.

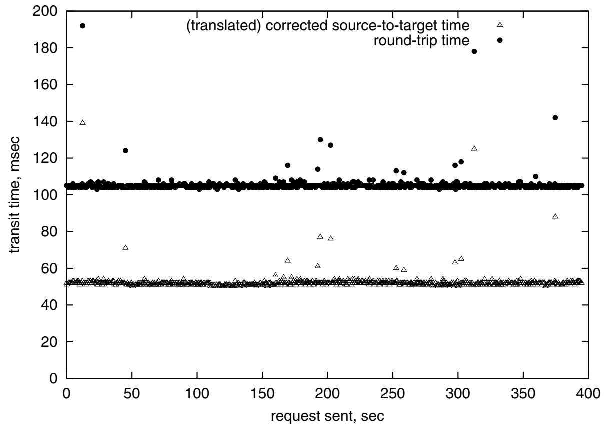

Fig. 17. After `fixclock` correction.

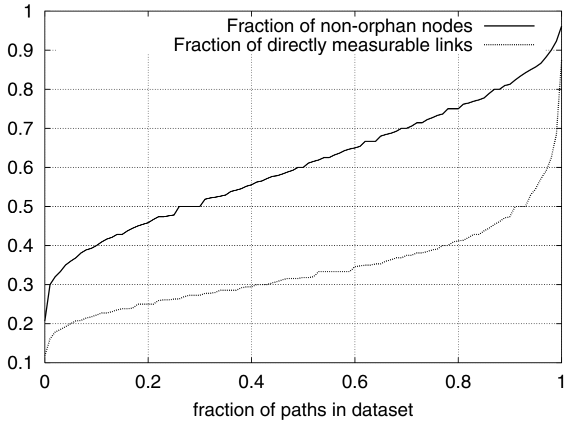

Fig. 15. Fraction of directly measurable links vs. fraction of non-orphan nodes in UPENN dataset

so, at the very least, must be expected to have some offset[6] from the source clock, denoted by $O_{x,y}$. The main problem is demonstrated in Figure 16. The target clock's timestamps may reflect rate-skew and jumps compared to the source clock. In this example, the source-to-target OTTs climb about 245 msec gradually over about 65 seconds, then drop back suddenly to begin the same linear rise again. This cannot reasonably be attributed to networking delays. It is clearly target clock behavior. On the other hand, we see stray OTT points that are indeed most reasonably attributed to network delays, not to clock behavior. This is corroborated by the generally coincident variation in round-trip times, which we know to be independent of the target clock, and so must indicate delay variation in at least one of the directions. The task of timestamp correction is to attribute the *apparent* delay variation to actual delay variation and to clock behavior: we want to remove target clock behavior artifacts from the target clock's timestamps. Once this is done, yielding data as depicted in Figure 17, we can proceed with consideration of the OTT variation, as discussed above, where we pretended that the only clock problem was an unknown offset.

In the case of this example, it is quite clear what the remote clock is doing, so correcting for it seems straightforward. The problem is less trivial when we attempt to formulate a fairly general algorithm to do the correction. Again, if we wish to handle only clock skew, or only jumps, and we are prepared to assume that congestion delays do not occur at inopportune times to obscure the target clock behavior, the problem is not too daunting. But we are faced with the general case, of an unknown combination of non-negligible skew, skew variation, jumps, and congestion delays, and we need to automate the correction efficiently in order to process thousands of probe data sets.

Several approaches to the detection of clock behavior, to allow the correction of timestamps, have been proposed. We particularly note Paxson's insights [19] regarding jump detection and regarding the statistical signature of non-negligible skew; and the convex-hull-based strategies of Zhang, Liu, and Xia [20]. Neither the Paxson nor the ZLX methods are able to address the variety of phenomena that confront us *simultaneously*. We have preferred to develop our own approach with a view to greater generality—an unavoidable objective in our research context—and to greater conceptual unity and simplicity than we have found in the previously proposed methods. Our `fixclock` algorithm, unlike its predecessors, does not begin by trying to locate the discontinuities in the target clock's behavior. Instead, it builds regions that are convincingly *free* of target clock discontinuities, regions throughout which the algorithm has grounds to assume that the target clock is behaving linearly. Each such region is subject to a common linear correction. The regions may cover all or very nearly all of the data, in which case it is likely that one could characterize the transitions between the regions as jumps or skew changes that occurred at precise times. But if the algorithm leaves gaps between the regions, typically as a result of obscuring congestion delays, this is of no concern: we are interested, in the first place, in the corrected timestamps, not

[6] As noted above, we need not discover the value of $O_{x,y}$, and, in fact, it is impossible to determine the value from one-way timestamps. We choose a value for display purposes, that brings the OTTs into range.

a precise history of the remote clock's behavior.

A detailed discussion of the fixclock algorithm, as well as background remarks on the Paxson and ZLX approaches, may be found in [21].

F. Network load and security issues

For experimental purposes, the network and router processing resources required by cing is not expected to be a concern. Resource-conscious choice of the number of network nodes participating in measuring paths is nevertheless important to avoid biased results. Wider use of any method that involves probing routers is also likely to consume significant resources at intermediate routers and therefore needs caution.

The use of our tool to investigate queuing delays on the links in a small number of paths will not impose a large load on the network. Similarly, it will not excite security software. In contrast, large scale measurements of thousands of paths will likely raise alarms, because ICMP Timestamp and ICMP Echo messages are often used for malicious purposes such as scanning and fingerprinting of remote systems for mounting appropriate attacks. The intensity of our experiments, as well as the need for picking random paths, which involves probing a large number of random IP addresses, further adds towards a suspicious looking traffic signature. We expected (correctly) that we would cause alarms when observed by firewalls or intrusion detection systems.

We therefore initially chose to embed a URL in our probe packets, pointing to a Web page explaining the nature of these probes. The Web page was visited several times, and there were a handful of routine complaints made through email to abuse@seas.upenn.edu (fewer than 5 over the course of our experiments) and two requests to filter out specific IP prefixes. For ensuring uninterrupted experiments and to make the innocent nature of our probes more obvious, we were advised to advertise appropriate host-names in DNS (e.g. netmap-X-contact-telnumber-email-at.cis.upenn.edu).

IV. Future Work

We know of several situations where cing can give inaccurate estimates. Our measurements seem to indicate that such situations are rare. However, although we have investigated on the order of 10,000 paths, we have used fewer than 10 sources. Consequently we are not confident that our study is comprehensive. Fortunately, we believe that we understand how to compensate for the inaccuracies and limitations that we know about. We have not incorporated them into cing because we felt that without compelling evidence that they would be useful, we could not justify increasing the complexity of our tool. Nor have we yet subjected these new techniques to testing comparable to the basic cing techniques. Should broader experience with cing indicate that these potential problems are limitations in practice, we would push forward in those areas.

For example, accuracy may be reduced when the two packets in a single packet-pair experience different latencies on their shared path. By sending both packets in a packet-pair to the "head" of the link we can get a sample of the distribution of packet-pair distortions. If we assume such distortions (whether due to differences in queuing delay or variations in time to generate ICMP packets) are independent of the queuing delay, then we can deconvolve our measured delay with the packet-pair difference in order to recover from packet-pair errors. This will not correct for (hypothetical) bias that may be *correlated* with large delay, though.

We *are* pursuing improving the coverage of our direct method. For instance, we can estimate the distribution of queuing delay in the shared overlap of overlapping segments, by using deconvolution in an adaptation of the indirect inference technique over non-tree topologies. This would allow us to measure any link, or segment, as long as neither of the endpoints of the segment were orphans. The value of such an extension is shown in Figure 15, although one would expect some decrease in accuracy and robustness of the resulting estimates, when compared to a pure direct approach. One could extend it further, to even cover orphans, by using TTL-expired messages to estimate delays on links anchored by orphans and combining these estimates with overlapping segments using deconvolution. As shown, these are less accurate than using than Timestamp measurements, but an improvement over no measurements at all.

In general, we are considering interesting hybrids of direct techniques and indirect techniques. The direct measurements improve the accuracy and reduce the computation cost, while the indirect measurements increase coverage. Our ongoing work along these lines is described in [16].

V. Summary and Concluding Remarks

We have revisited the possibility of measuring network-internal delay distributions using direct methods. Our work shows that the required infrastructure is deployed on the vast majority of routers (over 97%) and on *every* router in about half of all paths. We generally divide a path up into a small-number of multi-link segments, so the ability to investigate delay on *every* individual link in only half the paths is not much of a limitation in practice, and the infrastructure is in place to use cing almost anywhere. Thus, *direct measurement of network-internal delays without adding any extra measurement infrastructure is feasible.*

The accuracy of cing depended upon two broad assumptions. First, we assume that back-to-back packets experience the same performance and behavior on their shared path — we can then use the difference in time to measure the time spent on the unshared segment. Second, we assume that, with respect to queuing delay, our ICMP Timestamp probes behave indistinguishably from TCP or UDP packets. If so, we can then claim that measurements derived from our ICMP packets apply to other packets, too.

Our measurements show that the first assumption is true most, but not all, of the time. Back-to-back packets appear to experience only negligible differences in queuing delays, however ICMP packet generation times can sometimes be variable within a single router. Such variation may be misinterpreted as queuing delay. Nevertheless, our measurements show that more than 90% of all routers introduce such distortion less than 10% of the time. Further, if it becomes necessary, there are techniques that can recover from this distortion with high probability. We have avoided such techniques so far, because they increase the complexity of `cing` for very little gain.

Our measurements also show that despite the fact that our probes are addressed directly to the last router and not to the end node, and despite the fact that they are processed in the slow path rather than forwarded in the fast path, the excess overhead is roughly constant (other than a small amount of variability introduced during the generation of ICMP packets). Constant overhead cancels out because we subtract out the shortest measured time from all observations. The only variability that remains are queuing delays in the net.

Given that these assumptions are true, then it is clear from inspection (and confirmed by simulation [10]) that `cing` will return an accurate estimate of the distribution of queuing delay on each segment that it measures. Thus, *direct measurement of queuing delay using only existing infrastructure is accurate.*

Earlier [10] we had shown that direct techniques to measure network-internal queuing delays are more accurate than indirect inference techniques. Here we have shown that *one-way timestamps are a more accurate direct measurement technique than RTT measurements of probes designed to induce TTL-expired messages*. The advantage arises because of the existence of both asymmetric routes and asymmetric congestion on symmetric routes.

Routing irregularity in the Internet *does* often prevent the current version of `cing` from measuring the queuing delay on certain specific links. Fortunately, `cing` can detect (and report), in advance, its inability to measure specific links in isolation. Further, breaking up a path into a few multi-hop segments is usually sufficient to isolate congestion points. If the inability to measure specific links due to routing irregularity becomes a problem in practice, then, as mentioned in Section IV, we are pursuing several promising approaches to increase coverage, and improve accuracy, but at the expense of more complexity in `cing`.

ACKNOWLEDGMENTS

This work was supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795.

We thank John C. Parker, Dave Millar and Charles Buchholtz at UPenn, C. Ricudis at AUTH, A. Hatzistamatiou and G. Portokalidis at UCH, H. Bos and V. Viagers at LIACS, M. Hogsett, P. Lincoln at SRI, and C. Powell and J. Feigenbaum at Yale for making it possible to take measurements from different sites. We also thank S. Ioannidis, B. Knutsson, J. M. Smith and the members of the Distributed Systems Laboratory for valuable comments and suggestions throughout the course of this work.

REFERENCES

[1] V. Jacobson, "Congestion avoidance and control," in *Proceedings of Symposium on Communications Architectures and Protocols (SIGCOMM '88)*, 1988, pp. 314–329.

[2] ——, "pathchar - A tool to infer characteristics of Internet paths," available from *ftp://ftp.ee.lbl.gov/pathchar*, April 1997.

[3] A. B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proceedings of ACM SIGCOMM*, September 1999, pp. 241–250.

[4] A. Adams, T. Bu, R. Cáceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. Moon, V. Paxson, and D. Towsley, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 152–159, May 2000.

[5] N. G. Duffield and F. Lo Presti, "Multicast inference of packet delay variance at interior network links," in *Proceedings of IEEE INFOCOM*, vol. 3, April 2000, pp. 1351–1360.

[6] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Network delay tomography from end-to-end unicast measurements," in *Proceedings of the 2001 International Workshop on Digital Communications 2001 - Evolutionary Trends of the Internet*, September 2001.

[7] N. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280–292, June 2001.

[8] M. J. Luckie, A. J. McGregor, and H.-W. Braun, "Towards improving packet probing techniques," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2001*, November 2001, pp. 145–150.

[9] K. G. Anagnostakis, M. B. Greenwald, and R. S. Ryger, "On the sensitivity of network simulation to topology," in *Proceedings of the Tenth IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, October 2002, pp. 117–126, Fort Worth, Texas.

[10] K. G. Anagnostakis and M. B. Greenwald, "Direct measurement vs. indirect inference for determining network internal delays," *Performance Evaluation*, vol. 49, no. 1-4, pp. 165–176, September 2002, (Performance '02).

[11] ——, "On the feasibility of network delay tomography without infrastructure support," CIS Department, University of Pennsylvania, Tech. Rep. MS-CIS-01-35, December 2001.

[12] V. Paxson, "End-to-end routing behavior in the Internet," in *Proceedings of ACM SIGCOMM*, August 1996, pp. 25–38.

[13] V. Jacobson, "traceroute software," Original release available from *ftp://ftp.ee.lbl.gov/traceroute.tar.gz*.

[14] R. Govindan and V. Paxson, "Estimating router ICMP generation delays," in *Proceedings of the Passive and Active Measurements Workshop (PAM)*, March 2002.

[15] K. G. Anagnostakis, M. B. Greenwald, and R. S. Ryger, "On the feasibility of using existing infrastructure to measure network-internal delays," CIS Department, University of Pennsylvania, Tech. Rep. MS-CIS-02-21, July 2002.

[16] K. G. Anagnostakis and M. B. Greenwald, "A hybrid direct-indirect estimator of network internal delays," CIS Department, University of Pennsylvania, Tech. Rep. MS-CIS-02-31, December 2002, submitted to the *4th Workshop on Passive and Active Measurements (PAM03)*.

[17] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive routing of long-lived IP flows," in *Proceedings of ACM SIGCOMM*, September 1999, pp. 215 – 226.

[18] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC1771, http://www.rfc-editor.org/, March 1995.

[19] V. Paxson, "On calibrating measurements of packet transit times," in *Proceedings of ACM SIGMETRICS*, June 1998, pp. 11–21.

[20] L. Zhang, Z. Liu, and C. H. Xia, "Clock synchronization algorithms for network measurements," in *Proceedings of IEEE INFOCOM*, June 2002, pp. 160–169.

[21] R. S. Ryger, "`fixclock`: removing clock artifacts from communication timestamps," Department of Computer Science, Yale University, Tech. Rep. YALEU/DCS/TR-1243, December 2002.