

# A Systematic Approach to Measure RTT for Proximity-based Applications

Zhiheng Wang   Amgad Zeitoun   Sugih Jamin  
*Department of Electrical Engineering and Computer Science,  
 The University of Michigan, Ann Arbor*  
 {zhihengw,azeitoun,jamin}@eecs.umich.edu

**Abstract**—Round-trip time (RTT) between hosts provides valuable information about host proximity. Path RTT plays a crucial role in several overlay network construction protocols, peer-to-peer applications, and proximity-based server redirection in such applications as multiplayer gaming and content distribution network. Despite this important role, there is no systematic study of the RTT measurement process that we know of, where the problems involved in the measurement procedure have been studied, e.g., measurement accuracy under path congestion. The RTT of a path is usually estimated as the minimum of a number of measurements. Furthermore, the number of probes needed for the measurement procedure ranges from a few probes to more than 200 probes in different services and applications. Given the potential for wide deployment of these services and applications, the bandwidth consumed in measuring path RTT becomes significant. In this paper, we study the RTT measurement process trying to pinpoint the main challenges in capturing the minimum RTT along a path and how to minimize bandwidth requirement while maintaining measured RTT close to minimum path RTT. Our analysis builds upon the RTT phase-plot used by Bolot [1]. We also develop and evaluate a set of measurement techniques using simulation and real Internet experiments.

**Keywords**— Distance Measurement, Round-Trip Time, Proximity-based Application.

## I. INTRODUCTION

Several Internet applications have adopted the round-trip time (RTT) between hosts as a basic tool to approximate distance and to estimate location of hosts on the Internet. Overlay network such as Content-Addressable Network (CAN) [2], peer-to-peer applications such as Napster [3], various multiplayer game servers such as Gamespy [4], end-to-end Internet distance estimation services such as GNP [5] and IDMaps [6], geographical IP mapping services such as GeoPing [7], and end-host multicast applications such as Narada [8] and HMTP [9] all rely on measured RTT as a fundamental tool to determine the location of an Internet host relative to one or more measurement points. To elaborate:

- CAN is a distributed indexing application that builds an overlay network addressed by the indices. Routing on the overlay network is accomplished by hashing on the indices. The addressable space (range of index values) is di-

vided into a number of *zones*. Each zone is hosted by one or more physical nodes. Hosting nodes determine which zone they are in by measuring the RTTs to a given set of *landmarks*, which are then ordered according to their RTTs in ascending order. The ordering of the landmarks determines the zone a node is in. Inaccurate RTT measurements may result in landmark ordering that eventually lead to longer search time through the address space.

- Napster is a peer-to-peer file sharing application. When a Napster user searches for a file, Napster returns a list of peers hosting the requested file. The RTTs between the user and each peer are listed alongside. The user would then try to download first from the peer with the smallest RTT.

- Gamespy is an online gaming “lobby” where players can meet other players interested in playing one of a number of games. To use the Gamespy service, a player downloads and runs its client application. For each game a player is interested in, the Gamespy client lists existing servers and other players with their corresponding RTTs. Gamespy clients continuously monitor the RTTs between players.

- GNP is a distance estimation service that constructs a coordinate system to estimate distances between hosts. To determine the *coordinate* of a host, the host measures its RTTs to a set of *Landmark* machines (not the same landmarks as CAN’s) distributed across the Internet. These measurements are then used to position the host in a geometric space. In this space, the end-to-end distance between hosts is estimated by their geometric distance.

- IDMaps provides a distance estimation service by placing a set of measurement machines called *Tracers* on the Internet. Each Tracer measures its RTTs to various hosts on the Internet as well as to other Tracers. These measurements are used to build a *virtual* topology of the Internet, where the RTTs represent the cost (distance) of each link. End-to-end distance between any pair of IP addresses can then be computed on this virtual map.

- GeoPing locates an Internet host in a geographic region by mapping network delay to geographic distance. To estimate the geographic location of a host, GeoPing measures the RTTs between that host and a number of probe machines distributed across the Internet. The geographic lo-

cation of the host is then estimated by triangulation based on these RTTs.

- Narada and HMTP are examples of protocols to build end-host multicast overlay network. Both use measured RTTs between hosts to restructure the overlay network for improved performance. Each member of a multicast group periodically measures its RTTs to a handful of other members and adjusts the overlay network as necessary based on these measurements. The authors of [10] show the importance of adjusting the overlay network to the underlying physical network’s conditions to achieve good performance for a teleconferencing application running on top of the overlay network.

Most of these applications and services rely on measuring minimum path RTT for the correct operation of their services. The minimum path RTT is a property related to the topology of the network rather than the load of the network. This quantity is of most interest to applications and services that are sensitive to the topology of the network, such as GeoPing, GNP, IDMaps, CAN, and end-host multicast overlay applications. Other applications that use RTT for peer selection, such as Napster and Gamespy are interested in capturing the load on the network. Nevertheless, minimum path RTT still provides a valuable information for topological selection of different peers. Despite the great role of path RTT, there is no consensus among different applications and services on the RTT measurement process. For example, to measure the minimum RTT of a path, GeoPing sends 10–15 probes along the path. On the other hand, according to [5], GNP sends 220 probes to estimate the minimum path RTT. GeoPing and GNP measure the minimum RTT of each path once and use the measured result for an unspecified amount of time. Narada, HMTP, and Gamespy measure the RTT of each path periodically, with a 10 sec period in Narada’s case. Napster measures path RTTs between peers every time a user searches for a file. Not only is there a lack of consensus in RTT measurement frequency, inter-probe interval is also left unspecified in most cases. Thus despite the heavy reliance on measured RTTs we see in these applications and services, there has been no published results that systematically study the RTT measurement process.

While ad-hoc approaches to measuring RTT may be sufficient in proof-of-concept prototypes, a more systematic approach that minimizes the bandwidth requirement of the measurement process while still providing measured RTT close to the minimum RTT is required for sustainable large-scale deployment of these services. For example, if a large number of hosts try to compute their coordinates in GNP, the Landmark machines could be inundated with measurement packets given the way RTTs are

currently measured in GNP. If GeoPing or IDMaps measurement machines send a large number of probes to hosts on the Internet, they could trigger false security alarms at the destination networks. Widespread deployment of peer-to-peer applications that rely on periodic RTT measurements could use up a large portion of network bandwidth just for doing RTT measurements. Our main focus in this paper is to first try to identify and understand the main challenges and difficulties in capturing the minimum RTT along a path. Then we try to design some techniques to overcome those difficulties. Finally, we evaluate our technique through real experiments on some paths on the Internet.

We briefly summarize several related work in Section II. In Section III, we present the RTT measurement process, identify the main challenges in capturing the minimum RTT along a path, and furnish a set of techniques to detect and overcome these challenges. In Section IV, we present our experiment setup and the settings of the parameters involved in measuring the minimum RTT of some paths we studied on the Internet. We evaluate our set of techniques on the RTT samples captured from these paths. We also present and evaluate a method to differentiate between delay changes due to path changes and those due to congestion. In Section V, we present and correlate two methods to measure the RTT of a path. We summarize our minimum RTT measurement process in Section VI. Finally, we conclude in Section VII.

## II. RELATED WORK

A large number of studies on packet delay have been presented in the literature, e.g., [11], [1], [12], [13], [14]. To our knowledge, however, no previous studies of packet delay and loss on the Internet have reported the appropriate number of probes and the required frequency to capture the minimum RTT of a path. Mukherjee in [13] reported on a study in which he collected round-trip delay measurements for three Internet paths over a 24-hour period. He used ICMP ping to send 10 probes every 1 or 2 minutes with an inter-probe interval of 16.7 ms and 1 sec. One of the study’s important findings is that the distribution of RTT along a path can be approximately modeled by a shifted Gamma distribution. Bolot in [1] reported on a careful study of packet delay and loss characteristics observed on a single Internet path. He sent a group of 32-byte UDP packets over a 10-minute period with an inter-probe interval ranging from 8 ms to 500 ms. He also introduced a phase-plot technique as a method to analyze packet delays. We adopt this phase-plot technique in our analysis for the self-interference detection described in Section IV-B and the congestion detection technique described in Sec-

tion III-B.1. Acharya and Saltz conducted a large scale study to characterize the round-trip delay on 90 different paths [11]. Over a 48-hour period, they sent 64-byte ICMP packets to different hosts with an inter-probe interval of 1 second. Their main observations are: the probability of capturing minimum RTT of a given path, within a short period of time is high (a similar observation was also reported in [1], [15]), RTT distribution is skewed to the left with a long tail where mode RTT is very close to the minimum, and a large number of RTTs are within a window of 10 ms or 10% of the mode. The authors of [16] found that a router’s processing delay can reach values more than 1 ms (a delay of 35 ms has been observed in their study). The minimum delay within a hop can be observed at any time and does not depend on the router load. In agreement with previous studies, they also found that queueing delay distribution has a long tail and is well modeled by a Weibull distribution.

All of the above mentioned studies focused on RTT as a characteristic of Internet paths. While each study carefully documented how the authors went about measuring path RTT, it was not their intention to study the optimal method to measure path RTT as part of the normal running of an application. In contrast, our focus in this paper is on the RTT measurement process itself.

Aside from the above mentioned works, RTT measurement has also been studied in relation to TCP [17]. Under various variants of TCP, RTT is used to estimate retransmission timeout and to govern data transmission for the next transmission window. In contrast, our measurement process is intended for applications that use measured RTT for longer term decisions, such as peer or server selection. These decisions usually are re-evaluated once per session instead of per transport window. Thus while TCP updates the RTT estimates continuously for every data packet sent, our measurement process operates at a lower frequency.

### III. THE RTT MEASUREMENT PROCESS

The RTT of a stationary path, that is a path whose route does not change, can be formulated as:

$$RTT = t + p + q + \varepsilon, \quad (1)$$

where  $t$  is the transmission delay,  $p$  the propagation delay,  $q$  the queueing delay, and  $\varepsilon$  some random delay due to media access contention, router processing overhead, etc. (see Section III-B.1 for further discussion on  $\varepsilon$ ). The fixed component of the delay is  $RTT - q - \varepsilon$  (henceforth,  $minRTT$ ). If one takes a series of RTT measurements of a stationary path, assuming that at least one of the measurement sees an empty queue, the  $minRTT$  value of the path

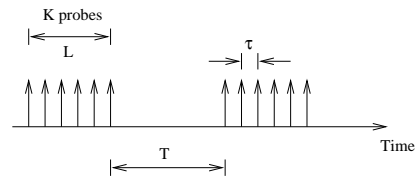


Fig. 1. Measurement process.

can be extracted by taking the minimum of the measurements.

As illustrated in Fig. 1, the RTT measurement process consists of sending  $K$  number of probes with inter-probe interval of  $\tau$  time units. We call this a *probe\_group*. Usually, the probe packet size is small to minimize resource usage in terms of processing and transmission delays. To ensure freshness of the measured distance, *probe\_groups* are sent  $T$  time units apart. While we want to send as many probes as necessary, at the highest frequency to achieve reasonably accurate measurements, we also want to avoid perturbing the measured system.

There are a set of challenges that governs the process of measuring the path minimum RTT. First, we try to identify those difficulties as:

**Self-interference:** When the inter-probe delay  $\tau$  is too small, the rate at which probe packets are sent may be high such that they will queue up behind each other at a given router causing self-interference that may eventually overloads the network. One has to take care of the rate at which probes are sent to avoid self-interference.

**Persistent congestion:** If the queues along the path experiencing congestion, it may happen that all probe packets sent during the measurement period  $L$  experience queueing delays in addition to the propagation delay. Therefore, the captured value of the RTT does not represent the minimum RTT of the path.

**Route change:** Another challenge that may cause a change in the measured minimum RTT is due to route change between hosts. Detecting route changes by reconstructing the end-to-end path from advertised route updates is not a viable option for most applications. Doing periodic traceroute to detect route changes is not only time consuming, but also takes up a lot of bandwidth. Furthermore, when the route between two hosts changes, it does not necessarily mean that the RTT between the two hosts also changes. Since most applications measure RTTs to detect changes in RTT, not route changes, the measurement process should not rely on having access to routing or topological information. To minimize the effect of route changes on the measured minimum RTT, the choice of the measurement period  $L$  should be small enough such that the probability that a route changes during this period is small. More about our settings for different values of

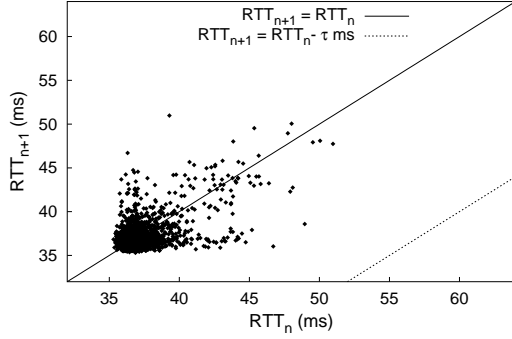


Fig. 2. An example of phase-plot.

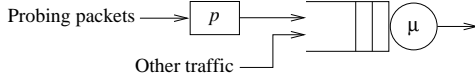


Fig. 3. Source-destination path model.

the measurement process parameters is presented in Section IV-A.

In the next subsections we address some techniques to overcome these difficulties in measuring the path minimum RTT.

#### A. Self-interference

Probe packets sent at small inter-probe delay  $\tau$  may cause self-interference due to the probe packets being queued behind each other in a router queue. The first thing is to verify that the choice of  $\tau$  is not too small. The phase-plot technique described in [1] provides a convenient way to study the queuing effects of the probe packets: given a set of RTT measurements, the phase-plot shows the correlation between two adjacent measurements. In a phase-plot, the  $x$ -axis ticks represent  $RTT_n$  and the  $y$ -axis ticks represent  $RTT_{n+1}$ , where  $n$  and  $n+1$  are indices of consecutive probes. Fig. 2 shows an example of a phase-plot.

Consider a path modeled as a single queue as shown in Fig. 3. The constant  $p$  represents the propagation delay of the path, the single queue with a finite buffer represents the variable component of the delay, and  $\mu$  the service rate. For probing packets that see an empty queue, the experienced delay for each probe is  $RTT_{n+1} = RTT_n = p + t + \varepsilon$ . If the queue is empty, experienced delays are clustered around the  $(p + s/\mu)$  (propagation delay plus transmission delay for a packet of size  $s$ ) point on the bottom left corner of the phase-plot. On the other hand, if a probe packet experiences queuing delay, the corresponding data point in the phase-plot will deviate from  $(p + s/\mu)$ . More specifically, if two or more adjacent probes are queued behind each other, then  $RTT_{n+1} = RTT_n - \tau$ , where  $\tau$  is the inter-probe delay. This shows up in the phase-plot as the line  $Y = X - \tau$ , the dashed line in Fig. 2 marks this

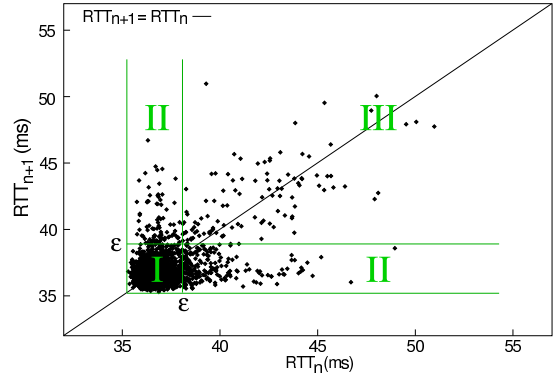


Fig. 4. Different regions in the phase-plot.

line. If the path is not congested and the inter-probe delay is large enough such that probe packets are not queued behind each other, the phase-plot of the RTT measurements should not contain points that lie on the  $Y = X - \tau$  line.

#### B. Route Changes and Persistent Congestion

As stated previously, the other challenges for measuring the minimum RTT along a path is the route changes and persistent congestion. In this section we look at how to detect whether a change in measured  $minRTT$  is due to route change or due to persistent congestion on the path. Different applications may want to react differently when the change in  $minRTT$  is due to one or the other of these causes. For example, distance and location services such as GeoPing, GNP, and IDMaps, which depend on the topological information of the hosts, may want to filter out changes due to congestion.

##### B.1 Congestion Regions and Congestion Components

When the  $minRTT$  of a path changes even when the forward and backward routes of the path have not changed, then we attribute this to the failure of the measurement process described in Section III to capture  $minRTT$  in the presence of persistent congestion on the path. To present our analysis on this failure, we first need to introduce our definitions of *congestion regions* and *congestion components*.

Consider again the phase-plot [1] presented in Section III-A. We divide the phase-plot into three *congestion regions* as shown in Fig. 4, with the following analysis:

1. Region *I* contains probe pairs that see empty queues and experience  $minRTT$  plus minor random overheads  $\varepsilon$  due to media access contention, router processing overhead, etc.
2. Region *III* contains probe pairs that always see a queue. This is the region of *persistent congestion*.
3. Region *II* contains probe pairs where one of the probe experiences queuing delay but the other does not, i.e.,

there is a congestion state transition between adjacent probes. This is the region of *transient congestion*.

An important parameter in the definitions of the *congestion regions* is  $\varepsilon$ , the threshold delimiting the regions. Our criterion for determining the value of  $\varepsilon$  is that when a probe reports  $RTT - p - t - \varepsilon \geq 0$  (using the notations from Eq. 1), we can be sure that the probe has seen a queue. We present our techniques in setting the value of  $\varepsilon$  in Section IV-D.

Given a trace of RTT measurements over a path, to detect the congestion level of the measured path, we compute the distribution of data points in the three regions of the phase-plot. We next define the *congestion component*,  $C_j$  as:

$$C_j = \frac{\text{Number of points in region } j}{\text{Total number of samples} - 1}, \quad (2)$$

where  $j = I, II$ , or  $III$ , and  $C_I + C_{II} + C_{III} = 1$ .

Therefore, the congestion component  $C_{III}$  can be used to detect whether a change in measured  $minRTT$  is caused by persistent congestion along a path. If the path experiencing a persistent congestion and the majority of probe packets are queued at some routers, then the percentage of data points in the phase-plot at  $C_{III}$  region would be high. That would not happen if the change in  $minRTT$  is due to route change along the path. We have observed this phenomenon during our experiment described in Section IV.

### C. Effect of Buffer Overflow

Persistent congestion may cause packet drop at different routers. This effect may be simplified by a buffer overflow, where a probe packet arrives at a router does not find a place in the router's queue. Therefore, probe loss should be used as an indication of persistent congestion. To study the effect of persistent congestion and queueing on  $C_{III}$  and to study the effect of buffer overflow on  $C_{III}$ , we decide to run some simulations where we have more control over the queueing and loss rate of a path. Fig. 5 shows our simulation setup. The bandwidth of the link between nodes  $G0$  and  $G1$  is 1.5 Mbps, the rest are 100 Mbps. The one-way propagation delay from  $G0$  to  $G1$  is 50 ms, and the one-way propagation delay for the other links is 2 ms. The random overhead  $\varepsilon$  is simulated by adding a random noise uniformly distributed between 0 and 2 ms. To study the effect of different loss rates, we experiment with different queue sizes at node  $G0$ . For each queue size, we run a simulation lasting 30 seconds.

We use 10 ON/OFF traffic sources, denoted as nodes labeled  $B0$  to  $B9$  in the figure, with Pareto distributed ON and OFF times to generate background traffic. Superposition of ON/OFF sources with Pareto distributed ON and

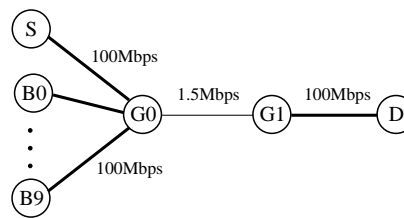


Fig. 5. Simulation setup.

OFF times generates aggregate traffic with long-range dependency [18]. Traffic on the Internet has been characterized as long-range dependent [19]. The average throughput of each Pareto source is 1 Mbps with a packet size of 512 bytes. The Pareto scale and shape parameters for the distribution of ON times are 51.2 packets, sent at peak rate, and 1.1 respectively; for OFF times, they are 0.1 second and 1.1 respectively. The measurement packets are sent from node  $S$  as 64-byte packets at a fixed rate of 256 Kbps (inter-probe interval of 2 ms). *Probe\_groups* are constructed from 100 probes for a total of 150 groups. Inter-group interval is set to 0 so that we can continuously monitor the queue length at  $G0$ . The inter-probe interval is chosen to the scale of the simulation time.

Congestion components of the probes are computed once every 200 ms and correlated with the average queue length at node  $G0$ , which is also computed once every 200 ms. The correlation is done by matching the sequence numbers of probes at the end-points and the log kept at  $G0$ . Fig. 6 shows the correlation between the congestion components  $C_{III}$  and the *average* queue length at  $G0$ . The figure shows that  $C_{III}$  tracks the average queue length very well up to the point of queue overflow. When queues are almost full, packets are either dropped or they have to wait for a queue-full of packets to clear before they see service. All probes that are not dropped thus see the same queueing delay and show up in the phase-plot in congestion region  $C_I$ . (Note that the definition of congestion region is relative to the smallest RTT of a trace.) This can be seen in Fig. 6(a), (b), and (c). At maximum queue length,  $C_{III}$  drops precipitously.<sup>1</sup> Fig. 6(d) shows the simulation with queue length of 10,000 packets (practically infinite), and we do not see such drop in  $C_{III}$  because the queue is never full. We thus augment the use of  $C_{III}$  to detect persistent congestion with detection of lost packets. We do not rely solely on packet losses to indicate persistent con-

<sup>1</sup>If we compute congestion components over intervals longer than queue occupancy time, we will be able to capture the actual  $minRTT$  of the path and the value of  $C_{III}$  will not drop precipitously. However, as pointed out in [20], queueing delay on the Internet has a long tail, with a strong peak at 65 sec. Hence we do not try to adjust the interval over which we compute congestion components by the expected queue occupancy time.

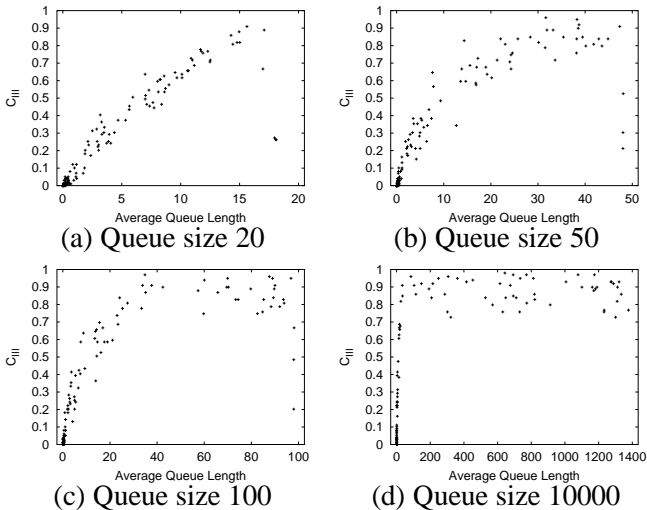


Fig. 6. Correlation between  $C_{III}$  and the average queue length for various queue sizes.

gestion because packet losses only occur when the queue overflows, while we are also interested in persistent non-empty queue that has not overflowed. Furthermore, packet losses can also occur during transient congestion.

In summary, when the RTT measurement process observes a change in the  $minRTT$  of a path, it can determine the cause of the change by calculating the  $C_{III}$  of the *probe\_group*. If  $C_{III}$  is large, or  $C_I$  is large but packet loss is detected, the change in  $minRTT$  can be attributed to congestion. Otherwise, the change in  $minRTT$  is probably due to route change.

#### IV. EXPERIMENT

Now we turn to study our techniques on different paths of the Internet. The main goal is to evaluate our set of techniques and see how close our observations match real RTT measurements collected on the Internet. According to the study done by Paxson [21], it is hard to sample a set of paths that are representative of the Internet. Nevertheless, we try to diversify our set of paths to include relatively distant and nearby targets. We also include hosts connected through different link technologies to the Internet. We collected our RTTs between 7 different hosts located in California (CA), Michigan (MI), and Virginia (VA) in the US and Sweden in Europe. The hosts in MI, VA, and Sweden were located at educational sites, the host in CA was located at a commercial site. Three of the hosts located in MI were connected through a Cable Modem technology to the Internet. The hosts in CA and MI were used as sources to measure RTTs to the hosts in CA, VA, and Sweden. In our first data set, we probe each path for a period of 24 hours in December 2000. For the duration of the experiment, we ran `traceroute` from both the source and the target hosts once every five seconds to detect route changes.

Our definition of a route change is strict, meaning that a path is considered stable if and only if all routers from the source to the target and from the target to the source did not change and were in the same order. We found no route changes both on the forward (source-to-target) and the backward (target-to-source) paths during our experiments. More data were collected between the same set of hosts in February 2001, December 2001, and January 2002.

#### A. Parameter Settings

We now discuss the setting of different RTT measurement parameters shown in Fig. 1, i.e.,  $\tau$ ,  $K$ ,  $L$ , and  $T$ .

First, for the measurement period  $L$ , we want a short enough period over which we would most likely not experience a route change. We did a literature search to determine the frequency of route changes on the Internet. In 1993, Chinoy [22] analyzed routing information exchanged between Autonomous Systems (ASs) and observed that 97% of networks experienced less than 10 reachability transitions in a period of 12 hours (a *reachability transition* is when a network that was reachable becomes unreachable, or vice versa), about 40% of the networks had only a single reachability transition during the 12-hour period. In 1997, Labovitz et al. [23] analyzed similar routing exchange data gathered over several months and observed that 80% of Internet routes are stable. Paxson et al. [20], [24] further conducted a large-scale study of end-to-end Internet routing dynamics and found that more than 87% of routes remained stable for a period of hours, less than 2% of paths experienced a route change on the order of 10 minutes, while all paths remained stable on the order of 60 seconds. Based on these results, we decided that 60 seconds would be a good upper bound on the value of  $L$ .

Paxson reported in [20] that queueing delay variations occur primarily on the time scales of 100 ms to 1 second, with a long tail. The queueing delay variations reported, however, do not capture state transition in the case of transient congestion. Our measurement process is targeted towards operating systems that provide a user-mode clock granularity of at most 10 ms. To allow for the operating system's scheduling overhead, we decide to set the inter-probe interval to a value not smaller than 20 ms. Thus 20 ms is the smallest value of  $\tau$ , which sets the upper bound on our probing rate.

In the subsequent sections we investigate whether we can use a smaller  $L$  and a larger  $\tau$ . Nevertheless, we first collected some measurements on the Internet using the upper bound values, i.e.,  $\tau = 20$  ms,  $L = 60$  sec, hence  $K = 3,000$  probes. Setting the value of  $T$  depends on

TABLE I

AVERAGE DEVIATION, IN MILLISECONDS, FROM  $minRTT$  CAPTURED WITH  $\tau = 20$  ms FOR EACH PATH WITH DIFFERENT INTER-PROBE INTERVALS (95% CONFIDENCE INTERVAL).

Path	$\tau = 80$ ms	$\tau = 320$ ms	$\tau = 1.28$ sec	$\tau = 2.56$ sec	exp.(mean=2.56 sec)
CA-VA	$0.04 \pm 0.008$	$0.09 \pm 0.014$	$0.17 \pm 0.018$	$0.22 \pm 0.025$	$0.2 \pm 0.023$
CA-Sweden	$0.02 \pm 0.007$	$0.07 \pm 0.009$	$0.13 \pm 0.02$	$0.15 \pm 0.023$	$0.13 \pm 0.02$
MI-CA	$0.04 \pm 0.008$	$0.1 \pm 0.013$	$0.2 \pm 0.02$	$0.29 \pm 0.041$	$0.26 \pm 0.03$
MI-VA	$0.06 \pm 0.012$	$0.14 \pm 0.017$	$0.27 \pm 0.022$	$0.33 \pm 0.026$	$0.29 \pm 0.027$
MI-Sweden	$0.04 \pm 0.009$	$0.15 \pm 0.087$	$0.4 \pm 0.35$	$0.49 \pm 0.35$	$0.33 \pm 0.18$
MI-Sweden(no outlier)	$0.04 \pm 0.009$	$0.11 \pm 0.016$	$0.22 \pm 0.02$	$0.31 \pm 0.025$	$0.24 \pm 0.023$

a lot of other factors and trade-offs. Since our study is focused on the appropriate settings for the  $probe\_group$ 's parameters, i.e.,  $\tau$ ,  $L$ , and  $K$ , we do not investigate the settings of  $T$  in this study; for the present experiment,  $T$  is exponentially distributed with a mean of 10 minutes. The PASTA (Poisson Arrivals See Time Averages) property of the exponentially distributed  $T$  makes this experiment unbiased to any periodic behaviors that may be present [25].

### B. Minimum Inter-probe Interval ( $\tau$ )

The first thing we need to verify is that our choice of  $\tau = 20$  ms is not too small, i.e., whether our probes experienced self-interference along different paths.

We use the phase-plot technique to detect probes self-interference described in Section III-A. For all of our data sets, when the path measured is not congested, we do not see the RTTs of our probes falling on the  $Y = X - \tau$  line, where  $\tau = 20$  ms. This means that our inter-probe interval of 20 ms did not cause self-interference.

Once we have an upper bound on the rate of probing ( $\tau = 20$  ms), we want to determine whether a larger  $\tau$  value can capture the  $minRTT$  of a path without losing measurement accuracy.<sup>2</sup> To that end, we vary the value of  $\tau$  from 20 ms to 2.56 sec. We also consider an exponentially distributed  $\tau$  with a mean of 2.56 seconds. Conducting a separate measurement experiment for each value of  $\tau$  runs the risk of the measurement processes seeing different states of the network. Thus instead of collecting a different set of measurements for each value of  $\tau$ , we resample our original measurements, collected with  $\tau = 20$  ms, at different frequencies. Because our original data sets did not see any self-interference between probes, the resampling process is valid.

Recall that our data sets consist of measurements collected at the upper bounds of our parameter settings,  $\tau = 20$  ms,  $L = 60$  seconds, which resulted in  $probe\_group$  of size  $K = 3,000$  samples. To increase  $\tau$ , we simply re-

<sup>2</sup>We used the paths where the  $minRTT$  observed by each  $probe\_group$  did not change during the 24-hour period.

sample each  $probe\_group$  of  $K$  probes at the appropriate interval. Then to study the effect of larger  $\tau$ 's, for each  $probe\_group$ , we calculate the difference between the  $minRTT$  captured at the given value of  $\tau$  and the  $minRTT$  captured at  $\tau = 20$  ms. Finally, we average the differences in  $minRTT$ s across all  $probe\_groups$  collected for that path.

Table I shows the average deviation from  $minRTT$  captured at  $\tau = 20$  ms, with 95% confidence interval, of these differences for various values of  $\tau$ . For the last column of the table, the value of  $\tau$  was taken from a Poisson sampling with a mean of 2.56 sec, but cut off at the minimum value of 20 ms (quantization level), and the maximum value of 5.12 sec. As can be seen from the table, even for  $\tau = 2.56$  sec, which, for  $L = 60$  sec, means that  $K = 24$  probes, we still capture the minimum RTT of a path with an average deviation less than 1 ms.

There is a single  $probe\_group$  in the MI-Sweden path where the  $minRTT$  observed during the 1-minute probing period increased from 124 ms to 746 ms, due to persistent congestion. The "MI-Sweden (no outlier)" row in the table shows the average deviation and confidence interval for the MI-Sweden  $probe\_groups$  without that single outlier. From this experiment, we conclude that *as long as self-interference is avoided, and the path is stationary, capturing the minimum RTT of a path depends very little on the inter-probe intervals during the measurement*. A similar observation was made by Allman and Paxson in [17], that different methods to estimate the retransmission timer (RTO) in TCP did not depend on how often RTT measurements were made.

### C. Number of Probes ( $K$ ) and Measurement Interval ( $L$ )

In this section, we study the effect of the number of probes on capturing the  $minRTT$  of a path. For each value of  $\tau$ , we form a  $probe\_group$  from the re-sampled data, varying the number of probes per group from 1 to 20, incrementing by 1. Recall that the original  $probe\_group$  size with  $\tau = 20$  ms is  $K = 3,000$  probes. If the first  $K$  probes of a given original  $probe\_group$  see a loss rate

greater than or equal to 25% [21], we throw the re-sampled *probe\_group* away and consider the subsequent  $K$  probes re-sampled at the given  $\tau$ . This process is repeated until all probes within the original *probe\_group* are exhausted. If we fail to form a re-sampled *probe\_group* of  $K$  probes at a given  $\tau$  interval, we simply remove that group from our calculations.<sup>3</sup>

Next we study the deviation of *minRTT* captured by  $K$  number of probes spaced  $\tau$  interval apart from that captured by sending 3,000 probes at 20 ms apart. Let  $\text{minRTT}_\tau^K$  be the *minRTT* of  $K$  probes with inter-probe interval of  $\tau$  time units. For each  $\tau$  and  $K$  we first compute  $\text{minRTT}_\tau^K$ , then we compute the difference between  $\text{minRTT}_\tau^K$  and  $\text{minRTT}_{20\text{ms}}^{3,000}$  for each *probe\_group*.

Table II shows, for all paths in the December 2000 data set, the average deviation of *minRTTs*, measured by varying the number of probes  $K$  ( $x$ -axis) at various  $\tau$  intervals, from the *minRTTs* measured by 3,000 probes sent 20ms apart, with the 95% confidence interval. The table shows that the first probe often experiences higher delay than subsequent probes (first probe usually experience additional delays due to ARP queries and route lookups at different routers along the path). With the increase of the number of probes, the deviation of the measured *minRTT* decrease and diminishing return is shown after a small number of probes, e.g., 8–12 probes in Table II. From Table II, we also observed that for a given number of probes, larger inter-probe interval  $\tau$  results in more accurate *minRTT* measurement. For small  $\tau$ , probes are sent in a fashion where they are closer to each other so that the possibility for them to see the same queueing variation increases[21]. Combining with our observations from Table I, we conclude that *minRTT* measurement depends on the measurement period  $L$  and the using of larger inter-probe interval  $\tau$  can alleviate the effect of queueing variation. Similar observations can be made from the December 2001 data set, which is shown in Table III.

In summary, the observations so far convince us that *minRTT* of a stationary non-congested path can be captured by sending a few probes at a reasonable inter-probe interval. We address the effect of route changes and persistent congestion in the following sections.

<sup>3</sup>For the CA-VA data set, we removed three out of 25,920 re-sampled *probe\_groups* due to high losses; these three groups occurred during a single 1-minute interval, the loss rate for the whole 3,000 probing attempts during this interval was 18.5%. For the CA-Sweden data set, we also removed three out of 24,120 re-sampled *probe\_groups*; these three also occurred at a single 1-minute interval with overall packet loss rate of 12.7%. For the MI-CA data set, we removed 46 out of 26,640 re-sampled *probe\_groups*, which occurred at 3 different 1-minute interval with overall packet loss rates of more than 31.3%.

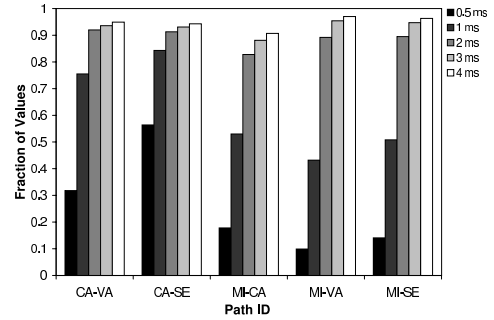


Fig. 7. Average fraction of RTTs within 0.5 ms, 1 ms, 2 ms, 3 ms, and 4 ms from the *minRTT* in each *probe\_group* over different paths.

#### D. Determining the value of $\epsilon$

Before we can study the effect of persistent congestion and route change on the computation of different congestion regions, we need first to determine the appropriate value for  $\epsilon$  which delimits the boundaries of different congestion regions. To that end, we conduct a statistical analysis on the RTT data described in Section IV. Table IV shows the average minimum, mean, and mode RTTs, with the 95% confidence interval, over all *probe\_groups* for each path. The mode RTT is closer to the *minRTT* than the mean RTT (apparently, as RTT distribution is known to have a long tail [11], [13], [16]). We next calculate the (*mode RTT* - *minRTT*) for each *probe\_group* and study the distribution of (*mode RTT* - *minRTT*) for each path. We find that the third quartile of this distribution is around 1 ms or less (last column in Table IV). We find this to be true for all the paths under study, i.e., even though the MI-CA path has a *minRTT* of 61 ms, MI-VA 35 ms, and MI-Sweden 125 ms, the third quartile of the difference distributions of these paths are 0.9 ms, 1.1 ms, and 1 ms respectively. Therefore, we set the value of  $\epsilon$  equal to double the value of the third quartile of (*mode RTT* - *minRTT*) distribution,  $\epsilon = 2 * (\text{mode RTT} - \text{minRTT})$ . For the paths under study that are connected to the Internet via fast Ethernet connections,  $\epsilon$  is thus 2 ms. For the hosts connected through the Cable Modem technology, we estimated the value of  $\epsilon$  to be 4 ms. Fig. 7 shows the fraction of RTT measurements within 0.5, 1, 2, 3, and 4 ms from the minimum RTT. The figure shows that, for all of our data sets for hosts connected through fast Ethernet connections, more than 80% of the RTTs measured are within 2 ms of the minimum value. This further justify our selection of  $\epsilon$ .

#### E. Effect of Persistent Congestion

In this subsection we illustrate the effect of persistent congestion and the use of our congestion component to differentiate persistent congestion from route changes.

TABLE II

AVERAGE DEVIATION, IN MILLISECONDS, FROM  $minRTT$  CAPTURED WITH  $\tau = 20\ ms$  FOR EACH PATH WITH DIFFERENT INTER-PROBE INTERVALS AND NUMBER OF PROBES (95% CONFIDENCE INTERVAL) FOR THE DEC. 2000 DATA SET.

Number of Probes (K)	$\tau = 40\ ms$	$\tau = 80\ ms$	$\tau = 1.28\ sec$	$\tau = 2.56\ sec$	exp.(mean=2.56 sec)
$K = 1$	$3.00 \pm 2.16$	$2.94 \pm 2.12$	$2.53 \pm 1.96$	$2.53 \pm 1.96$	$4.34 \pm 3.18$
$K = 4$	$2.08 \pm 2.05$	$2.24 \pm 2.14$	$1.20 \pm 1.09$	$1.31 \pm 1.31$	$1.62 \pm 1.49$
$K = 8$	$2.02 \pm 2.11$	$1.31 \pm 1.61$	$1.05 \pm 1.09$	$1.10 \pm 1.21$	$1.36 \pm 1.05$
$K = 12$	$1.68 \pm 1.84$	$1.23 \pm 1.61$	$0.97 \pm 1.09$	$1.01 \pm 1.21$	$1.21 \pm 1.42$
$K = 16$	$1.21 \pm 1.61$	$1.19 \pm 1.61$	$0.93 \pm 1.09$	$0.97 \pm 1.21$	$0.87 \pm 0.99$
$K = 20$	$1.43 \pm 1.69$	$1.07 \pm 1.45$	$0.89 \pm 1.09$	$0.32 \pm 0.07$	$0.99 \pm 1.29$

TABLE III

AVERAGE DEVIATION, IN MILLISECONDS, FROM  $minRTT$  CAPTURED WITH  $\tau = 20\ ms$  FOR EACH PATH WITH DIFFERENT INTER-PROBE INTERVALS AND NUMBER OF PROBES (95% CONFIDENCE INTERVAL) FOR THE DEC. 2001 DATA SET.

Number of Probes (K)	$\tau = 40\ ms$	$\tau = 80\ ms$	$\tau = 1.28\ sec$	$\tau = 2.56\ sec$	exp.(mean=2.56 sec)
$K = 1$	$3.46 \pm 0.67$	$3.51 \pm 0.70$	$3.41 \pm 0.67$	$3.44 \pm 0.67$	$3.46 \pm 0.76$
$K = 4$	$1.79 \pm 0.39$	$1.82 \pm 0.44$	$1.60 \pm 0.33$	$1.56 \pm 0.30$	$1.46 \pm 0.26$
$K = 8$	$1.39 \pm 0.35$	$1.35 \pm 0.34$	$1.10 \pm 0.16$	$1.04 \pm 0.14$	$1.11 \pm 0.17$
$K = 12$	$1.23 \pm 0.33$	$1.24 \pm 0.34$	$0.95 \pm 0.14$	$0.89 \pm 0.11$	$0.92 \pm 0.11$
$K = 16$	$1.09 \pm 0.31$	$1.15 \pm 0.34$	$0.83 \pm 0.11$	$0.80 \pm 0.10$	$0.85 \pm 0.11$
$K = 20$	$1.04 \pm 0.31$	$1.08 \pm 0.33$	$0.78 \pm 0.10$	$0.75 \pm 0.10$	$0.79 \pm 0.11$

TABLE IV

THE AVERAGE MINIMUM, MEAN, AND MODE RTTs FOR ALL *probe\_groups* ON EACH PATH WITH THE 95% CONFIDENCE INTERVAL. THE LAST COLUMN IS THE THIRD-QUARTILE OF THE (*mode RTT*-*minRTT*) DISTRIBUTION.

Path	min (ms)	mean (ms)	mode (ms)	$3^{rd}$ Quartile of (mode - min) (ms)
CA-VA	$62.6 \pm 0.013$	$65.7 \pm 2.8$	$63.7 \pm 0.29$	0.7
CA-Sweden	$175.8 \pm 0.007$	$178 \pm 2.03$	$176.3 \pm 0.17$	0.4
MI-CA	$61.3 \pm 0.012$	$66.5 \pm 6.24$	$62.1 \pm 0.06$	0.9
MI-VA	$35.3 \pm 0.010$	$36.7 \pm 0.044$	$36.3 \pm 0.027$	1.1
MI-Sweden	$124.5 \pm 0.008$	$126.1 \pm 0.3$	$125.4 \pm 0.02$	1

We first show how the congestion component  $C_{III}$  can be used to detect whether a change in measured  $minRTT$  is caused by persistent congestion along a path. For this, we use a trace of RTT measurements conducted in February 2001 between our MI and VA sites. The measurement process was set up exactly the same as in Section IV-A. The dashed line labeled “ $minRTT$ ” in Fig. 8 (to be read with the  $y$ -axis on the left side) shows the  $minRTT$ s from one day’s worth of *probe\_groups*, each consisting of  $K = 3,000$  probes. Note that the  $minRTT$ s measured between 1 and 2 am and again between 9 and 10 pm on the 13th are much larger than the 35 ms recorded for the rest of the trace. From the traceroute collected throughout the trace, we determine that there was no route change either in the forward or backward route during these periods when  $minRTT$  deviated from the minimum value.

Studying the phase plot of the trace confirms that indeed the smallest RTTs measured during these times have

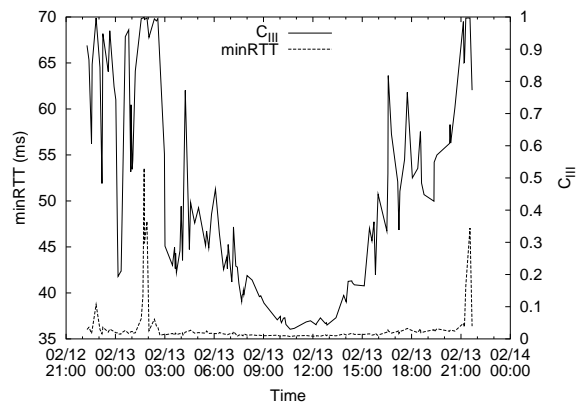


Fig. 8. Sample paths of  $minRTT$  (left  $y$ -axis) and  $C_{III}$  (right  $y$ -axis).

shifted from the minimum RTT of the whole trace. The solid line labeled “ $C_{III}$ ” in Fig. 8 (to be read with the  $y$ -axis on the right side) shows the value of congestion component  $C_{III}$ , which is the fraction of data points in con-

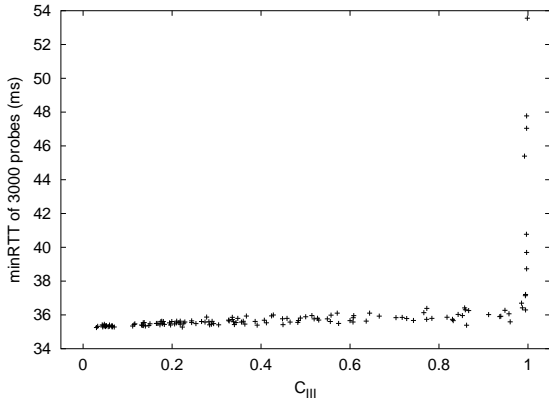


Fig. 9. Correlation between  $C_{III}$  and  $minRTT$ .

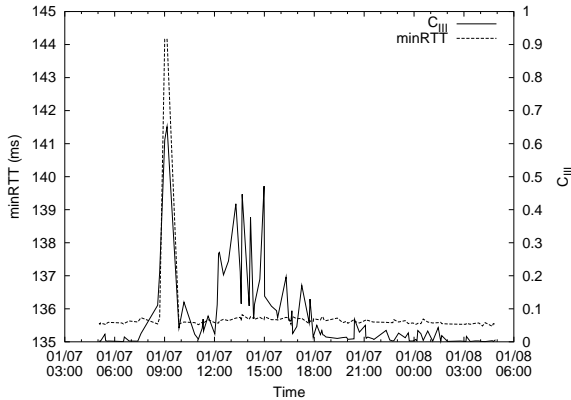


Fig. 10. Sample paths of  $minRTT$  and  $C_{III}$  for Sweden-VA.

gestion region III of the phase-plot. As can be seen from the figure,  $C_{III}$  reaches 1 during periods of persistent congestion. Fig. 9 shows the correlation between  $C_{III}$  and  $minRTT$  of this data set. From this figure, we see that when  $C_{III}$  approaches 1, the path is experiencing persistent congestion and the smallest measured  $minRTT$  is probably not the actual minimum RTT of the path.<sup>4</sup>

#### F. Effect of Buffer Overflow

Fig. 10 shows the  $minRTT$  and  $C_{III}$  of the CA-Sweden path collected in January 2002. There is a sharp increase in  $minRTT$  around 9 am on the 7th.  $C_{III}$ 's value is 0.6 at this point, which indicates some degree of congestion but not significant enough (i.e., it is still  $< 0.95$ ) to attribute the change in  $minRTT$  to persistent congestion. Our traceroute data does not capture any route changes in either direction. However, during this period we observe a high degree of packet loss rate ( $\geq 28\%$  loss rate). We conclude that this is indeed a period of persistent congestion, however the congestion component  $C_{III}$  fails to detect it due to massive queue buffer overflow. This observation verifies our simulation studies presented in Section III-C, where  $C_{III}$

<sup>4</sup>This observation also has implications on bandwidth measurement tools such as `pathchar` [15], such congestion effect would result in a much lower bandwidth measurement.

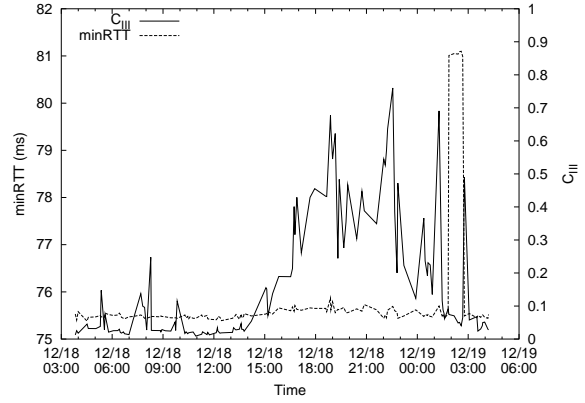


Fig. 11. Sample paths of  $minRTT$  and  $C_{III}$  for CA-VA

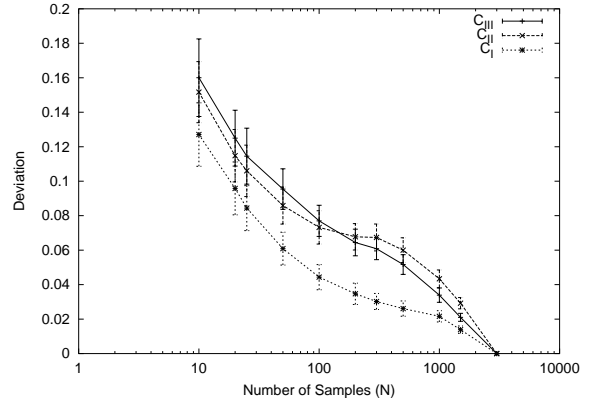


Fig. 12. The deviation of  $C_j^N$  from  $C_j^{3,000}$  for  $N \leq 3,000$ .

drops significantly when probe packets are dropped due to persistent congestion.

#### G. Effect of Route Change

We now show how the congestion component  $C_{III}$  can be used to detect when a change in measured  $minRTT$  is caused by routing changes. Fig. 11 shows the measured  $minRTT$  and the corresponding  $C_{III}$  for the CA-VA path in our December 2001 data set. The figure can be read in the same manner as Fig. 8. There is high variability in measured  $minRTT$  near the end of the trace. From our traceroute data, we determine that the path was experiencing routing changes during this time. The hop count for the path changes from 19 to 23, to 22, and back to 19 in a forty-five minutes period. Note that  $C_{III}$  during this period is low and we did not observe any severe packet losses, which tells us that the variability in  $minRTT$  is not due to persistent congestion, in agreement with the traceroute data.

#### H. Computation of Congestion Components

In this section we study how changing the number of samples effect the computation of congestion components. We resample our data set, reducing the number of probes,

$K$ , per *probe\_group* while keeping the same period  $L$ . Let the number of the samples be  $N$ , we define  $C_j^N$  as the  $C_j$  calculated with  $N$  samples. Fig. 12 shows  $|C_j^N - C_j^{3,000}|$  as we vary the number of samples. The figure is computed from the same data set used in Section IV-E.

Fig. 12 shows that  $C_j$  calculated from a small number of probes can still give a good approximation of  $C_j^{3000}$ , e.g., in our data set,  $C_{III}^{10}$  could be expected to have a deviation of 0.16 from  $C_{III}^{3000}$ . Also, we observed that when the number of probes increases beyond 100, we will have diminishing return on the accuracy improvement by increasing the number of probes. We note the  $X$ -axis in Fig. 12 is in log scale.

In Section IV-C, we have noticed that without persistent congestion, a small number of probes are capable of capturing the *minRTT* along a path. Hence a plausible measurement scheme would be sending out a small number of probes, e.g.,  $K=10$ , and calculate  $C_{III}^K$ . If  $C_{III}^K$  is small, we adopt the measured *minRTT* as the path *minRTT*. Otherwise we continue to increase  $K$  until  $K=100$ . During this process, if a small  $C_{III}^K$  is reached, we use the measured *minRTT* as the path *minRTT*. Should this is not true, i.e., until we increase  $K$  to 100 and  $C_{III}^{100}$  is still significant, persistent congestion could be claimed for this measurement period. In this case, we should stop sending more probes due to the ongoing path congestion. New measurement procedure should be started later on. The study on the accuracy and efficiency of such scheme will be our future study.

## V. MEASUREMENT TOOLS

Given the administratively decentralized nature of the Internet, the `ping` utility has turned out to be the common primitive to all extant RTT measurement tools. To measure the RTT between a source and a target host, `ping` sends an *Echo-Request* ICMP message from the source host to the target host, and measures the elapsed time until an *Echo-Response* ICMP message returns from the target host. We call a pair of such echo request-response packets a *probe*. Due to the recent heightened security awareness of network administrators [26], increasingly large number of routers and firewalls do not forward ICMP packets. Furthermore, some system administrators turn off support for ICMP “request-reply” in their operating system kernel. To address the disadvantages of ICMP ping, a new variant based on the hand-shaking of TCP connection establishment has recently been developed [27], [28]; we call this technique “TCP ping.” To measure the RTT between a source and a target host using TCP ping, the source host tries to open a TCP connection with the target host, i.e., the source sends a TCP SYN packet to the target.

The time elapsed until a TCP SYN-ACK or RST packet returns from the target is the RTT. We evaluate the advantages of using TCP ping over ICMP ping by conducting a simple experiment as follows. We send a group<sup>5</sup> of ICMP and TCP probes to 20,811 different hosts across the Internet. Of these, 534 hosts (2.6%) failed to report RTT by either methods—this may be due to connectivity or reachability problems [23]. Of the remaining hosts, 99.9% reported RTTs by TCP ping, while only 87.85% reported RTTs by ICMP ping; that is, more than 12% of our ICMP probes were filtered and dropped either by routers or the target hosts.

Upon receipt of an ICMP *Echo-Request* packet, the operating system kernel can immediately reply to it; whereas upon receipt of a TCP SYN packet, the kernel would first try to match or spawn a process to handle it, which causes extra delay in the resulting RTT measure. Although previous studies [27], [28] have used TCP ping to measure RTTs, we are not aware of any published studies comparing the RTTs measured using TCP ping against those measured using ICMP ping. To evaluate the correlation between the RTTs measured by TCP ping against those measured by ICMP ping, we run each from 4 source hosts to 3 sets of HTTP mirrors: 40 Akamai mirrors (<http://www.akamai.com>), 22 FreeBSD mirrors (<http://www.FreeBSD.org>), and 40 academic web sites worldwide. For each source-target pair, a group of TCP probes is sent in parallel with another group of ICMP probes. Probes in each group are spaced out at 2.56 seconds interval. Each ICMP group is shifted by 1.28 seconds from the TCP group, i.e., the first ICMP probe is sent 1.28 seconds after the first TCP probe, and so on. Groups are spaced out at exponentially distributed intervals with a mean of 10 minutes. We collected 48 hours worth of trace for each source-target pair.

We take the minimum RTT measured by each group of probes as the RTT of the path. Fig. 13 plots the RTTs reported by ICMP ping (on the  $x$ -axis) and the corresponding RTTs reported by TCP ping (on the  $y$ -axis) for Akamai, FreeBSD, and academic mirrors. The correlation coefficients of the Akamai, FreeBSD, and academic data sets are 0.994, 0.989, and 0.925 respectively. Fig. 13(d) is the correlation data for the academic sites after removing the 2 sites causing the outliers visible in Fig. 13(c). The correlation coefficient for Fig. 13(d) is 0.997. The loss rate seen in the traces for these outliers has a range of 6%–19.4% for both TCP and ICMP pings, while other sites experienced only 0.1%–4.4% loss rate; we thus attribute the cause of the outliers to congestion.

<sup>5</sup>8 probes per group

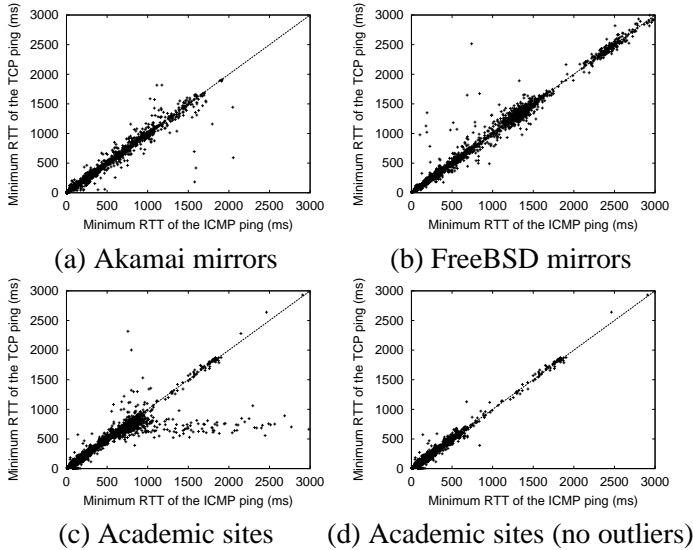


Fig. 13. Correlation between ICMP ping and TCP ping.

We conjecture that TCP ping numbers can be so highly correlated with ICMP ping numbers even though TCP probes require extra kernel processing because ICMP packets are often forwarded at a lower priority at routers. From these results, we conclude that the minimum RTTs captured by TCP pings can be highly correlated with those captured by ICMP ping if one is careful to identify and discount congested paths.

## VI. SUMMARY AND DISCUSSIONS

Informed by the observations and results presented in this paper, we now summarize our recommendations for techniques used to measure the minimum RTT along a path.

- Probe packets need to be small in size to minimize the transmission delay and processing time at each router along the path. The default packet size for TCP probes (40 bytes) or ICMP probes (84 bytes) is reasonably small enough to satisfy this condition.
- Although TCP probes may experience extra delay due to kernel processing time, we have shown that the kernel processing time is usually negligible compared to the path RTT. Furthermore, the advantages of TCP probe over ICMP probe make the TCP technique is more attractive.
- Since measured  $minRTT$  does not depend on  $\tau$ , the inter-probe interval  $\tau$  is set to a reasonable value that does not cause self-interference among probes. Using the phase-plot technique presented in Section III-A one can detect the self-interference of probe packets.
- For a stationary non-congested paths, few probe packets are required to yield a reasonable estimate for the path minimum RTT.

- The interval between *probe\_groups* ( $T$ ) depends on the application. Some applications and services need to measure the RTT only once (e.g., GeoPing), others need to periodically sample the path RTT (e.g., Gamespy, IDMaps, HMTp, and Narada). Following the PASTA principle, RFC2330 [25] recommends using an exponentially distributed  $T$ .

- The phase-plot technique presented in Section III-A and Section III-B provides a valuable information about the detection of probes self-interference and the detection of persistent congestion and route changes along the measured path.

The advantage of our measurement process to the network is clear; it reduces the amount of bandwidth put on the network by applications measuring RTTs between users. To the applications, our measurement process minimizes the number of packets the applications must generate and process to obtain accurate RTT of a path. Even for applications that require only a rough measure of path RTT, and therefore can afford to be sloppy in its measurement process, our paper shows that they only need to send a small number of probes, instead of 200 or more probes, to measure path RTT. Other network distance metrics such as bottleneck link bandwidth are currently measured as functions of path RTT (e.g., *pathchar* tool [15]). Hence we believe that our measurement process would also benefit applications interested in network metrics other than path RTT.

## VII. CONCLUSION

In this paper, we have presented a systematic approach to measure the RTT along a path. The RTT of a path is usually estimated as the minimum of a number of measurements. This number ranges from a few packets in some applications to more than 200 packets in some other services. Given the potential for wide deployment of these services and applications, the bandwidth consumed in measuring path RTT becomes significant. Additionally, recent heightened security awareness among network administrators may trigger false alarms if a large number of measurement probes are observed on the networks.

The main goal is to study the main difficulties in the RTT measurement process and to find different techniques to minimize bandwidth requirement while maintaining measured RTT close to minimum RTT of the path. To that end, we first identified the different challenges in measuring the minimum RTT along a path. Then we presented a set of techniques to overcome those difficulties. Finally, we have conducted a set of experiments to empirically study our measurement process on several Internet paths at different time-of-day and day-of-week.

We found that the RTT measurement process depends very little on the inter-probe interval. Our results show that usually sending a few measurement packets is sufficient to capture the minimum RTT along a stationary path.

We have also developed and evaluated a novel technique to determine our level of confidence in the measured *minRTT*. This technique is based on the use of phase-plot presented in [1]. By studying the phase-plot of a given *probe\_group* we can differentiate whether a change in *minRTT* is caused by route changes or persistent congestion. Also, using the phase-plot approach we can determine if the setting of the inter-probe delay caused the probes to experience a self-interference or not.

Applications using the RTT measurement process developed in this paper can reduce their bandwidth usage while maintaining good confidence in the measured RTT. With the introduction of more network-aware applications that adapt their behavior to network condition, the use of this mechanism can reduce overall bandwidth requirements on the Internet.

#### REFERENCES

- [1] J-C. Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet," *Proc. of ACM SIGCOMM '93*, pp. 289–298, Sep. 1993.
- [2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. of ACM SIGCOMM '01*, August 2001.
- [3] "Napster," <http://www.napster.com>.
- [4] "GameSpy Arcade," <http://www.gamespy.com>.
- [5] T. S. Eugene Ng and H. Zhang, "Towards Global Network Positioning," *ACM Internet Measurement Workshop 2001*, November 2001.
- [6] P. Francis et al., "IDMaps: A Global Internet Host Distance Estimation Service," *ACM/IEEE Transactions on Networking*, vol. 9, no. 5, Oct. 2001.
- [7] V. N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *Proc. of ACM SIGCOMM '01*, August 2001.
- [8] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. of ACM SIGMETRICS '00*, 2000.
- [9] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *Proc. of IEEE INFOCOM '02*, June 2002.
- [10] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture," *Proc. of ACM SIGCOMM '01*, August 2001.
- [11] A. Acharya and J. Saltz, "A Study of Internet Round-Trip Delay," Department of Computer Science Technical Report CS-TR-3736, University of Maryland, Dec. 1996.
- [12] K. Claffy, G. Polyzos, and H-W Braun, "Measurement Considerations for Assessing Unidirectional Latencies," *Journal of Inter-networking: Research and Experience*, vol. 4, no. 3, pp. 121–132, Sep. 1993.
- [13] A. Mukherjee, "On the Dynamics and Significance of Low Frequency Components of Internet Load," Tech. Rep. CIS-92-83, University of Pennsylvania, Dec. 1992.
- [14] D. Sanghi, O. Gudmundson, K. Agrawala, and B. Jain, "Experimental Assessment of End-to-End Behavior on Internet," *Proc. of IEEE INFOCOM '93*, pp. 867–874, Mar. 1993.
- [15] Allen B. Downey, "Using Pathchar to Estimate Internet Link Characteristics," *Proc. of ACM SIGCOMM '99*, pp. 241–250, Sep. 1999.
- [16] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot, "Analysis of Measured Single-hop Delay From An Operational Backbone Network," *Proc. of IEEE INFOCOM '02*, June 2002.
- [17] Mark Allman and Vern Paxson, "On Estimating End-to-End Network Path Properties," *Proc. of ACM SIGCOMM '99*, pp. 263–274, 1999.
- [18] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level," *Proc. of ACM SIGCOMM '95*, pp. 100–113, Aug. 1995.
- [19] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *ACM/IEEE Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
- [20] V. Paxson, "End-to-End Routing Behavior in the Internet," *Proc. of ACM SIGCOMM '96*, pp. 25–38, Aug. 1996.
- [21] V. Paxson, "End-to-End Internet Packet Dynamics," *Proc. of ACM SIGCOMM '97*, Sep. 1997.
- [22] B. Chinoy, "Dynamics of Internet Routing Information," *Proc. of ACM SIGCOMM '93*, pp. 45–52, Sep. 1993.
- [23] C. Labovitz, G.R. Malan, and F. Jahanian, "Internet Routing Instability," *Proc. of ACM SIGCOMM '97*, 1997.
- [24] Y. Zhang, V. Paxson, and S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput," Aciri technical report, May 2000.
- [25] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics," RFC 2330, Internet Engineering Task Force, May 1998.
- [26] Stefan Savage, "Sting: a TCP-based Network Measurement Tool," *USENIX Symposium on Internet Technologies and Systems '99*, Oct. 1999.
- [27] Sandra G. Dykes, Clinton L. Jeffery, and Kay A. Robbins, "An Empirical Evaluation of Client-Side Server Selection Algorithms," *Proc. of IEEE INFOCOM '00*, 2000.
- [28] Martin Horneffer, "Assessing Internet Performance Metrics Using Large-Scale TCP-syn Based Measurements," *Passive & Active Measurement Workshop (PAM'00)*, 2000.