Diagnosing Spatio-Temporal Internet Congestion Properties

Leiwen Deng and Aleksandar Kuzmanovic
Department of EECS, Northwestern University
2145 Sheridan Road, Evanston, IL, 60208, USA
Email: {karldeng,akuzma}@cs.northwestern.edu

***Abstract*— The ability to accurately detect congestion events in the Internet and reveal their spatial (where they happen) and temporal (how long they last) properties would significantly improve our understanding of how the Internet operates. In this paper we present *Pong*[1], a novel measurement tool capable of effectively diagnosing congestion events over short time-scales ($\sim$ 50 ms or longer), and simultaneously locating multiple congested points on an end-to-end path at the granularity of a single link.**

***Pong* (i) uses queuing delay as indicative of congestion, and (ii) strategically coordinates router-targeted probes with end-to-end probes. Moreover, it (iii) has small measurement overhead, (iv) achieves high sampling frequency by sending probes to *all* intermediate nodes, including un-congested ones, (v) dramatically improves spatial detection granularity (*i.e.*, from path segments to individual links), by using short-term congestion history, (vi) considerably enhances the measurement quality by adjusting the probing methodology (*e.g.*, send 4-, 3-, or 2-packet probes) based on the observed path topology, and (vii) deterministically detects moments of its own inaccuracy. We conduct a large-scale measurement study on over 21,000 Internet paths and present their spatio-temporal properties as inferred by *Pong*.**

I. Introduction

Motivation. Despite existing measurements of spatial (*e.g.*, [2], [3]) and temporal (*e.g.*, [4]) Internet congestion characteristics, developing a more insightful and *joint spatio-temporal* congestion "picture" of today's Internet is valuable for several reasons. First, obtaining answers to questions such as (i) where does congestion occur (*e.g.*, at the edges, in the core, between ASes), (ii) how long and how intensive is congestion depending on its location, or (iii) what is the probability of observing one or more congested points on an end-to-end path, would significantly improve our understanding of the Internet. Moreover, the ability to accurately pinpoint congested locations in real time is useful for fault diagnosis (*e.g.*, [3]), for the design of advanced delay-based congestion control protocols (*e.g.*, [5], [6]) or for building enhanced distributed monitoring systems (*e.g.*, [7], [8]).

Congestion has several appearances. The most severe one induces packet losses, occurring whenever no buffering is available for newly arriving packets at a node. However, the first indicative of congestion in the Internet is increased *queuing delay*, occurring whenever the arrival traffic rate is greater than the link capacity. A variable queuing delay leads to jitter which can hurt the performance of interactive applications or delay-based congestion control protocols. Queuing is more likely to be observed; as such, it provides more diverse and much finer-grain information about the underlying congestion. Hence, we focus on the increased queuing delay as a viable indicator of congestion.

Contributions. Our contributions are threefold. First, we develop a novel measurement methodology capable of accurately inferring spatio-temporal congestion properties of Internet paths. We demonstrate that our methodology achieves high precision in both spatial and temporal domains. Second, we implement our methodology and build a novel measurement tool — *Pong*. Due to its lightweight (low-overhead) nature, *Pong* can be used not only for on-demand measurements, but for long term monitoring as well. Third, we conduct a large-scale measurement study (over 21,000 Internet paths), provide novel insights and enhance our understanding of spatio-temporal Internet congestion properties.

Methodology. Consider two endpoints and the path between the two, consisting of a number of intermediate nodes. Assume a symmetric path for the moment for simplicity. The goal is to detect the spatio-temporal congestion property of this path: at which links does congestion happen and how long does it last?

Our methodology consists of two steps. First, endpoints strategically combine end-to-end probes with probes targeted to a certain intermediate node (*e.g.*, 4 packets in total per sample measurement) to estimate queuing delays on the two half paths between that intermediate node and both endpoints. Despite the fact that the resulting set of equations is under-constrained [9], and thus the unique solution does not exist, we manage to develop tight *lower and upper bounds* for the queuing delays on targeted half paths. Second, we sequentially repeat the above coordinated measurements to *all* nodes along the path. By correlating half-path estimates that are close in space and time, we manage to locate congestion at the granularity of a single link.

Challenges and Approach. A challenge for the proposed methodology is path asymmetry: packets on forward and backward paths between two endpoints do not necessarily share the same set of intermediate nodes [10]. The same holds for path segments traveled by TTL-limited probes or by probes "targeted" to intermediate nodes [3]. To solve the problem, we introduce *measurable pairs* — pairs of nodes on forward and

A subset of this work appears in the Proceedings of ACM SIGMETRICS 2007 [1]. This work is supported by a Cisco Collaborative Research grant.

[1]The Pong source code as well as our measurement data sets are publicly available at http://networks.cs.northwestern.edu/pong/.

backward paths that could be effectively exploited to support the coordinated measurements used by our approach. The key insight is that even if probe packets do not travel the same path segments, the 4-packet (*4-p*) methodology is still viable as long as probes share the same *congested* segments.

In certain scenarios, even when probes share different congested points, we are able to exploit them by demoting the approach to use 3- or 2-packet coordinated probes. While the queuing-delay bounds that we develop in such scenarios are necessarily looser than in the 4-packet scenario, they are still extremely valuable and extensively used in practice. Indeed, unique to our methodology is its *self-adaptability*. We leverage detected path topology properties and inferred congestion patterns to automatically select the appropriate probing technique (*i.e.*, 4-, 3-, or 2-packet).

Handling Measurement Inaccuracies and Interferences. Our methodology combines both end-to-end probing and probing to intermediate routers. The former is considered susceptible to several interference factors, *e.g.*, clock skew and route changes. The latter is considered susceptible to other factors, *i.e.*, some routers may assign a low priority to such probes or not respond to them at all. As we will demonstrate in this work, we are still able to effectively handle all these measurement interferences in a *unified and methodical way*: we define a new measure, *quality of measurability*, which helps *Pong* to detect moments of its own inaccuracy, as we explain in more detail below.

In our methodology, we detect moments of all these interferences, including: router response anomalies (*e.g.*, low priority, ICMP queuing, rate limiting), clock skew and clock jumps, temporary route alterations, permanent route changes, other unknown interferences that introduce extra packet delays. Once interferences are detected, we address them by either filtering out measurement data affected by the interferences or by annotating measurement results with error estimates.

Qualify of Measurability. The techniques we use to adaptively choose a probing technique and the approach we use to detect measurement interferences exploit a single measure – *Quality of Measurability (QoM)* (Section III-C.1). The long term average value of *QoM* indicates how well a path condition is satisfied, hence how well a probing technique can be applied. The instantaneous value of *QoM* depicts the quality of a measurement sample. It is also a sensitive indicator of anomalies. When *QoM* shows a small instantaneous value, we infer that the corresponding probing packets are affected by some anomaly. On the contrary, if *QoM* shows a large value, we know that the sample has a good quality.

Evaluation and Measurement. We evaluate *Pong* in the Emulab testbed, and show that our tool exhibits desirable measurement performance. It gives high resolution and accuracy both in identifying multiple forward congestion points and in decoupling backward congestion points. We also show that it quickly converges to a suitable probing technique in response to changing network conditions.

Next, we deploy *Pong* on the PlanetLab testbed [11]. At first, we perform self-consistency tests on a large number of paths and verify that congestion events at links shared by different paths are successfully detected by *independent* endpoint pairs. Because *Pong* has a small measurement overhead (*i.e.*, $\approx 18Kbps$ per path), it is suitable for large-scale Internet measurements. We measure over 21,000 Internet paths in an attempt to reveal their spatio-temporal congestion properties.

Our measurement results show that the phenomenon of "heavily" congested edges is actually dominated by congestion on intra-AS links. At the same time, we show that inter-AS links behave almost the same at edges and in the core. Due to the "bipolar" behavior of intra-AS links, the Internet edge is 4.5 times more congested than the core. Our time-domain analysis indicates that congestion happening at edges is clustered in time while congestion happening in the core is relatively dispersed. Finally, we quantify the probability to observe multiple congestion points on a path over a given time interval; we find that it grows as a power function of the time interval length, and decays exponentially with the number of congested points.

Avenues for Future Research. We are currently building a triggered measurement system based on *Pong* on the PlanetLab platform [11]. The goal is to explore potential correlation between congestion and other factors, *e.g.*, whether congestion events exhibit certain spatial locality, or whether congestion behavior is correlated to the peering policies between ISPs. The system exploits the lightweight nature of *Pong* to perform long term link congestion monitoring. Some congestion events being captured on a given link will serve to trigger the system to monitor the set of links that is in the "neighborhood" of the given link. The underlying idea is to select vantage points that provides the highest measurement quality for objective links.

The reminder of this paper is structured as follows. In Section II, we introduce our measurement methodology, while in Section III, we present our implementation — the measurement tool *Pong*. Then, in Section IV, we summarize our measurement results from large scale Internet experiments. Next, in Section V, we present related work. Finally, in Section VI, we conclude.

II. Methodology

A. Coordinated Probing

The first step of our approach is to estimate queuing delays on *unidirectional* half paths between endpoints and an intermediate node. Because queuing can happen on both forward and backward directions of a path, it is impossible to directly measure queuing delays of targeted unidirectional half paths. Thus, we develop several novel methods to measure them indirectly.

1) **Four-packet Probing***:* Consider a simplified symmetric path scenario shown in Figure 1. Denote by f a sample one-way delay between the source and the destination measured by a "forward" probe. Next, denote by f_{min} the minimum one-way delay observed (among measured samples.) Finally, denote by Δf the corresponding *queuing delay* experienced by the "forward" probe, and computed as $\Delta f = f - f_{min}$. Likewise, denote by b a sample one-way delay between the destination and the source measured by a "backward" probe, and compute the corresponding backward queuing delay as $\Delta b = b - b_{min}$. Next, denote by s a sample *round-trip*

Notations or Terms	Brief Description	Section
measurable point	An intermediate node that replies to TTL-limited probes	II-A.1
f, b, s, d probe	Four types of probing packets	II-A.1
Δf, Δb, Δs, Δd	Queuing delays measured by the four types of probing packets	II-A.1
Δf_s, Δf_d	Queuing delays of unidirectional half paths (forward direction)	II-A.1
pairing	To pair up an *s* and a *d* probe	II-A.2
measurable pair	A pair of intermediate nodes used for pairing	II-A.2
L_{fs}, U_{fs}, L_{fd}, U_{fd}	Lower and upper bounds of Δf_s, Δf_d	II-A.3
P_x, P_{fs}, P_{fd}	Congestion probability of half paths	II-A.7
weighted congestion count	A measure quantifies congestion extent on a specific path segment	II-B.2
weighted congestion frequency	A measure quantifies congestion status on a detected congestion point	II-C

TABLE I

SUMMARY OF IMPORTANT NOTATIONS AND TERMS

delay experienced by a "source" probe, from the source to the middle-point M and back, as indicated in Figure 1.

Similarly to the above scenarios, compute the corresponding queuing delay as $\Delta s = s - s_{min}$. In a real implementation, Δs is the cumulative queuing delay experienced by a TTL-limited packet sent by the source, and the corresponding ICMP reply packet generated by router M. Finally, using a "destination" probe in the same way as the "source" probe, we compute the *round-trip* queuing-delay estimate from the destination to *the same* router M and back, as $\Delta d = d - d_{min}$.

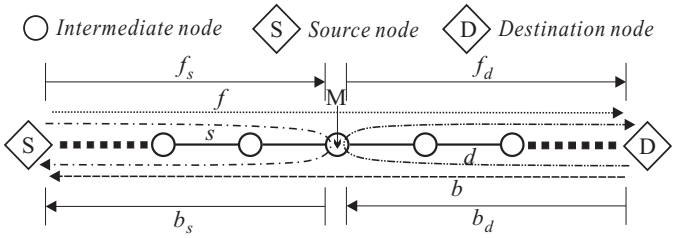


Fig. 1. 4-p probing: a symmetric path scenario

Assume a *coordinated* effort by the *sender* (*i.e.*, source node) and *receiver* (*i.e.*, destination node) such that corresponding probes travel the appropriate segments *nearly* synchronously (we define "nearly" below). This is easy to achieve in the case of f and s probes, both of which are generated by the sender. Provided that the two probes are sent back-to-back, they will travel the half path from S to M concurrently [3]. The same holds for b and d probes over the D to M half path. The challenging part is to provide nearly concurrent travels for the f and d probes over the M to D half path; and the same for the b and s probes over the M to S half path. This difficulty exists because those probes are generated by different nodes. To deal with this problem, we apply coarse-grained endpoints synchronization and probing-time rescheduling (Section III-B.2).

It is essential to understand that contrary to capacity and available bandwidth estimation techniques (*e.g.*, [2], [12], [13], [14]) which *require* packets to be sent back-to-back to reveal desired network internals, we do not share the same problem here. Because our scheme relies on the *queuing delay* as the indicative of congestion, it is the stationarity of that value (on time scales of *tens to hundreds of ms* [4]) that effect our measurement accuracy. Second, even when desynchronization between probes grows to a level that jeopardizes the inference accuracy, we are able to accurately detect and filter such events, as we explain later in Section III-E.2.

Now, denote by f_s the *one-way* delay on the half path from S to M, and by Δf_s the queuing delay on the same half path. Next, denote by Δf_d the queuing delay on the half path from M to D; similarly, denote by Δb_d the queuing delay on the half path from D to M, and by Δb_s the queuing delay on the half path from M to S. Finally, denote by c the clock offset between endpoints S and D. Then, the following two sets of equations hold.

$$\begin{cases} f = f_s + f_d + c \\ b = b_s + b_d - c \\ s = f_s + b_s \\ d = f_d + b_d \\ f + b = s + d \end{cases} \quad or \quad \begin{cases} \Delta f = \Delta f_s + \Delta f_d \\ \Delta b = \Delta b_s + \Delta b_d \\ \Delta s = \Delta f_s + \Delta b_s \\ \Delta d = \Delta f_d + \Delta b_d \\ \Delta f + \Delta b = \Delta s + \Delta d \end{cases} \tag{1}$$

Given the measurements we obtain via *4-p* probing (Δf, Δb, Δs, and Δd), our goal is to estimate Δf_s, Δf_d, Δb_s, and Δb_d. Unfortunately, the set of equations (1) is underconstrained (*e.g.*, [9]) and thus the unique solution does not exist. However, we are capable of developing *lower and upper bounds* for targeted half paths. Indeed, our final goal is to understand if there is congestion on a link and what is its intensity. For such a cause, estimations of queuing-delay *ranges* are often sufficient.

For example, for Δf_s, it could be shown that $\Delta f_s \in [\Delta f - \Delta d, \Delta f]$ or $\Delta f_s \in [\Delta s - \Delta b, \Delta s]$. Thus, whenever Δb or Δd are small, we have a tight bound for Δf_s. Indeed, given that around 83% of links experience trivial congestion (Section IV-D.3), we are often able to reach tight queuing-delay bounds in practice. More details about the algorithm we use to develop queuing-delay bounds can be found in Appendix I.

2) **Measurable Pairs***:* The key challenge for the coordinated probing approach is that the forward and backward Internet paths between two endpoints could be, and often are, disjoint [10]. Thus, it is not guaranteed that probe packets will travel the same path segments, as idealized in Figure 1. To address this problem, we introduce *measurable pairs*. A measurable pair is any pair of nodes, typically one on the forward and the other on the backward path, for which probes share the same *congested* path segments. For example, in the *4-p* case, any pair of nodes that satisfies the condition

$$\Delta f + \Delta b \approx \Delta s + \Delta d \tag{2}$$

is a *4-p* measurable pair. Note that condition (2) is not strong in practice. Since we are measuring queuing delays instead of absolute delays, we only need probes to share those *congested* segments, not the whole half-paths.

Figure 2 illustrates an example scenario. Despite the fact that forward and backward paths are disjoint, it is still possible to find pairs of nodes for which condition (2) is satisfied. Later,

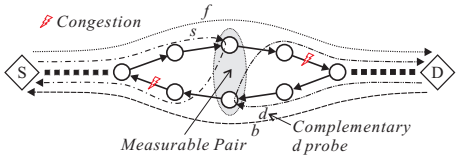


Fig. 2. *4-p* probing scenario

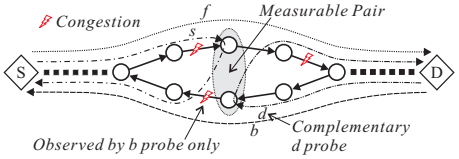


Fig. 3. *Fsd* probing scenario

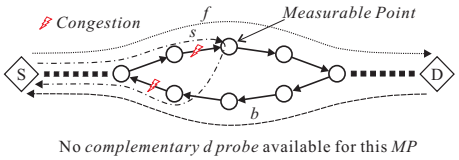


Fig. 4. *Fsb* probing scenario

in Section III-C.2, we provide an algorithm for effectively *pairing* s and d probes based on the *long term correlation* among delays measured by the four types of probes[2]. Also, our Internet experiments indicate that measurable pairs are quite stable over time. This is because Internet routes do not change so frequently, and also because *congestion patterns* and their spatial distributions are typically stationary over longer time scales, as we demonstrate later.

For a given intermediate node, we are not always able to find a measurable pair that satisfies condition (2). One example is given in Figure 3. Because b probe experiences congestion on a path segment not shared by either s or d probes, condition (2) is not satisfied. In such scenarios, we use *demoted* probing techniques consisting of a smaller number of packets (*e.g.*, 3 or 2 packets). Below, we explain such techniques and develop appropriate queuing-delay bounds.

3) **Fsd Probing***: Fsd* probing technique is similar to *4-p probing* to a great extent. It also requires pairing of *s* and *d probes*, but it does not require b probes. Any pair of nodes that satisfies the condition

$$\Delta f \approx \Delta s + \Delta d \tag{3}$$

is an *fsd* measurable pair. Figure 3 shows an example scenario. As long as f, s, and d probes share the same *congested* path segments, condition (3) is satisfied.

Denote by x an arbitrary half path and by L_x and U_x lower and upper queuing-delay bounds for that half path. For space constraints, consider path segments on the forward path only, f_s and f_d. Then, it could be shown that the half path queuing delays on the forward path become

$$\begin{cases} (\ \Delta f_s \in [L_{fs},\ U_{fs}]\ \ \Delta f_d \in [L_{fd},\ U_{fd}]\ ) \\ L_{fs} = max(0,\ \Delta f - \Delta d) \\ U_{fs} = min(\Delta s,\ \Delta f) \\ L_{fd} = max(0,\ \Delta f - \Delta s) \\ U_{fd} = min(\Delta d,\ \Delta f). \end{cases} \tag{4}$$

4) **Fsb Probing***: Fsb* probing technique employs *f*, *s*, and *b* probes. Since it does not use *d* probes, no pairing is required. The condition for *fsb* probing is:

$$\Delta s \approx \Delta f + \Delta b. \tag{5}$$

In this case, the half path queuing delays on the forward path become

$$\begin{cases} (\ \Delta f_s \in [L_{fs},\ U_{fs}]\ \ \Delta f_d \in [L_{fd},\ U_{fd}]\ ) \\ L_{fs} = max(0,\ \Delta s - \Delta b) \\ U_{fs} = min(\Delta s,\ \Delta f) \\ L_{fd} = max(0,\ \Delta f - \Delta s) \\ U_{fd} = \begin{cases} max(0,\ \Delta f - \Delta s + \Delta b), & \Delta s > \Delta b \\ \Delta f, & \Delta s \le \Delta b. \end{cases} \end{cases} \tag{6}$$

[2]We do not need to resolve aliases. Alias resolution is neither necessary nor sufficient to determine the pairing.

Figure 4 depicts an *fsb* probing scenario. Because congestion resides only on paths shared by f, s, and b probes, no *complementary d probe* is required. Condition (5) is satisfied, and *fsb* probing can work effectively.

5) **Two-packet Probing***:* Common for the above three techniques is the existence of measurable conditions, in which probes show certain correlation, as exemplified by Equations (2), (3), (5). Not always are such conditions required to estimate targeted half path queuing delays. Fs packet probing is one such example (bd is another one, which we omit for space constraints). In this case, only two types of probing packets are sent — f and s probes. Fs probing can always be used since it does not require any special condition. As long as an intermediate node responds to TTL limited probes, fs probing is applicable.

For fs probing, the half path queuing delays on the forward path become

$$\begin{cases} (\ \Delta f_s \in [L_{fs},\ U_{fs}]\ \ \Delta f_d \in [L_{fd},\ U_{fd}]\ ) \\ L_{fs} = 0 \\ U_{fs} = min(\Delta s,\ \Delta f) \\ L_{fd} = max(0,\ \Delta f - \Delta s) \\ U_{fd} = \Delta f. \end{cases} \tag{7}$$

6) **Choosing a Probing Technique***:* A same node might become eligible for multiple types of probing techniques at the same time. For example, the *fsb* scenario shown in Figure 4 could be "handled" by fs probing as well. However, the queuing-delay bounds are necessarily looser as the probing technique gets more and more demoted. For example, the U_{fd} bound for *fsb* probing (Equation (6)) is tighter than that for fs probing (Equation (7)), simply because b probe is available. Likewise, the U_{fd} bound becomes even tighter for *fsd* probing (Equation (4)), which is because the d probe potentially brings even more information about the underlying congestion status. Hence, we select the probing technique based on the following order of *priority*:

$$pri(4p) > pri(fsd) > pri(fsb) > pri(2p) \tag{8}$$

Finally, for a given node, probing techniques can change over time. This is either due to path changes or congestion location hopping. As a result, for example, a node might be used for *4-p* probing at first, but is demoted to *2-p* probing after some time, then switches to *fsd* or *fsb* probing from time to time, and is finally promoted back to use *4-p* probing.

7) **Inferring Congestion***:* Now we explain how to convert half path queuing delays to a more appropriate measure, the probability a half path is congested. Indeed, probability is a more appropriate measure because absolute values of queuing delay can be relative, yet persistence of congestion is what we are interested in and it is well represented by the probability measure. For a half path x, denote by P_x its congestion

probability. To compute P_x, we apply a simple threshold-based approach. If the *lower bound* of queuing-delay on a half path is greater than a given threshold[3] we set P_x to 1. Likewise, if the *upper bound* is below the threshold, we set P_x to 0. Otherwise, P_x takes a value between 0 and 1 which is calculated as a function of the *lower bound*, *upper bound* and the threshold. When P_x is greater than 0.5, we regard the half path as congested, otherwise not congested.

B. Locating Congestion Points

1) **Probe Scheduling***:* If we were concerned only by determining spatial path congestion properties, we would choose to send probes to *all* intermediate nodes *concurrently*. This would provide an accurate "snapshot" of a path at a given moment in time. However, such an approach would limit the frequency by which we can sample the path; consequently, it would degrade the minimum measurable time-scale of congestion. This is because some routers in the Internet rate-limit ICMP responses (that we use for s and d probes) to a relatively small rate (*e.g.*, one response per 500 ms or 1 sec [3]). Yet, congestion events can happen at shorter time-scales [4].

To improve the path sampling frequency, we adopt *sequential scheduling*. Assume n measurable points (*e.g.*, either individual nodes or measurable pairs) on an end-to-end path. At time t_0, we send the first coordinate probe set (*e.g., 4-p* probes) to the first measurable point. Next, at time $t_0 + 50\,ms$, we send another probe set (*e.g., 4-p* again) to the second measurable point, adjacent in space to the first one, *etc.* We probe each measurable point once per second. Thus, we are capable of improving the path sampling frequency by a factor of n relative to the concurrent probing approach described above. At the same time, it still ensures that probe sets to consecutive measurable points are sent relatively close in time (50 ms apart). This allows our *switch point* approach to accurately determine spatial congestion properties despite the fact that probe sets are not measuring network conditions at exactly the same moments.

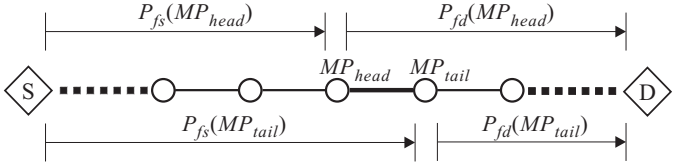


Fig. 5. Switch point approach

2) **Switch Point Approach***:* Consider a path segment (a link in most cases) between two consecutive measurable points (MPs). Denote by MP_{head} a measurable point closer to the source, and by MP_{tail} a measurable point closer to the destination, as illustrated in Figure 5. To become a *switch point*, the path segment must satisfy both of the following conditions:

$$P_{fs}(MP_{tail}) > 0.5 \; and \; P_{fd}(MP_{head}) > 0.5, \quad (9)$$
$$P_{fs}(MP_{head}) \leq 0.5 \; or \; P_{fd}(MP_{tail}) \leq 0.5. \quad (10)$$

Condition (9) means that the half path between source and MP_{tail} is congested *and* the half path between MP_{head} and destination is congested. Likewise, condition (10) means that the half path between source and MP_{head} is *not* congested OR the half path between MP_{tail} and destination is *not* congested. The above conditions are applicable for path segments that are not at edges. For the two path segments at edges, the following conditions are used: $P_{fs}(MP_{tail}) > 0.5$, for the first path segment, and $P_{fd}(MP_{head}) > 0.5$, for the last path segment.

To quantify the extent of congestion, we define a measure called *weighted congestion count* (*WCC*). Each time a path segment is identified as a *switch point*, we increase *WCC*. If there were no uncertainties, we would always set the increment to be 1 such that *WCC* works exactly like a counter. However, due to the existence of uncertainties, we use a weighted increment ($\in [0, 1]$) to capture our confidence on each switch point. For example, the tighter the estimates of half path queuing delays are, the more confident our conclusion is, hence the higher the weight will be. In addition, we *double* the increment amount if the following condition is satisfied for a switch point:

$$P_{fs}(MP_{head}) \leq 0.5 \;\; and \;\; P_{fd}(MP_{tail}) \leq 0.5. \quad (11)$$

Condition (11) is much stronger than condition (10), and when satisfied it indicates a much higher confidence of congestion at this switch point. We maintain *WCC* records of several different time scales. The shortest time scale is 10 seconds.

C. Tracing Congestion Status

The sequential scheduling approach "visits" each switch point once every second, while our aim is to detect congestion events at much shorter time scales. Indeed, we send consecutive probe sets $50\,ms$ apart. Thus, once candidate congestion points are located by the *switch point* approach, we can effectively *reuse* coordinated probes generated to *all* intermediate nodes to improve the sampling frequency of congested nodes.

The method is as follows. We examine *weighted congestion count* (*WCC*) history of each path segment to determine candidate congestion points. A *WCC* history consists of six 10-second *WCC* values of the recent 60 seconds. Based on this, we compute a *congestion positive rate (CPR)* ($\in [0, 1]$) as described in Table II. A path segment is considered congested if its congestion positive rate is nonzero.

	Condition satisfied	CPR assigned
1	WCC of recent 10 seconds $>$ 1.5	1.00
2	WCC of recent 20 seconds $>$ 2	0.90
3	WCC of recent 30 seconds $>$ 2.5	0.80
4	WCC of recent 40 seconds $>$ 2.75	0.65
5	WCC of recent 50 seconds $>$ 3.0	0.50
6	WCC of recent 60 seconds $>$ 3.25	0.35
7	None of the above is satisfied	0

TABLE II
COMPUTATION OF CONGESTION POSITIVE RATE (CPR)

Suppose we receive a probing result from a measurable point that says there is congestion in front of this measurable point (on the half path between the source and that measurable point) but no congestion behind it. Based on this, we can draw the conclusion that one or more candidate congestion points in front of this measurable point are congested at

[3]In Section III-E.1, we provide a self-adaptive algorithm for setting a threshold value based on path properties.

this moment. A special case, which we extensively use in practice, is that there exists *a single* candidate congestion point in front of this measurable point. For this case, we can deterministically tell that the candidate congestion point is congested at this moment. Because a considerable percent of links in the Internet are not congested (Section IV-D.3), we can achieve high sampling frequency of congested points by reusing probes to un-congested ones.

To quantify traced congestion status, we define a measure called *weighted congestion frequency* (*WCF*) for each path segment. *WCF* is computed over 30-second time intervals. If there were no uncertainties, the meaning of *WCF* would be the percentage of probe samples in the 30 seconds reporting that a specific path segment is congested. However, due to uncertainties, we introduce a weight ($\in [0, 1]$) for each sample when updating *WCF*. The weight is proportional to the congestion positive rate mentioned above.

III. Implementation

A. Challenges and Objectives

We implement a novel measurement tool — *Pong*. Like many other tools, *Pong* faces several practical challenges. The first is the lack of clock synchronization between end-hosts, and the second is the limitation of the maximum probing rate imposed by ICMP rate limiting at routers. We cope with the first problem by developing a simple clock skew disposal algorithm explained in Section III-E.2. Indeed, because we are only interested in measuring *relative one-way delays*, we simply follow the well-established procedure (*e.g.*, [5], [6]). Moreover, the ability to detect relative one-way delay errors by exploiting s and d probes in addition to f and b probes improves our measurement accuracy. We cope with the second problem by selecting long Internet paths thereby increasing the total number of measurable points along a path. As a result, the overall probing rate improves.

Pong is unable to measure very slight or transient congestion (*e.g.*, below 50 ms time scales), but rather measures congestion with considerable scale that repeats frequently over time. Thus, although the scope of measurements is inherently limited, *Pong* is able to detect relatively stable and longer-lived congestion events, which certainly enhances our understanding of spatio-temporal Internet congestion properties.

B. Probing

1) **Sending and Receiving Probes***:* All four types of probing packets are UDP packets to a same UDP port. Each packet is associated with a *probe id* and a *round id*. *Probe ids* differentiate probes sent to different measurable points along the path, and *round ids* differentiate probes sent in different measurement rounds. Each measurement round is one second long. During each round, sender and receiver sequentially send *sets* of probes to each of the intermediate measurable points, as explained above.

For *f* and *b* probes, we encode *probe id* and *round id* in the UDP payload. For *s* and *d* probes, we encode *probe id* and *round id* in the 16-bit *Identification* field of IP header. The size of all four types of probing packets is usually 56 bytes — a 28-byte header plus a 28-byte UDP payload. For *s* and *d* probes, the received packets are ICMP responses generated by intermediate nodes; their size is usually 56 bytes. Assume an end-to-end path with 10 measurable points. *Pong*'s measurement overhead in terms of bandwidth becomes $10 \times 4 \times 56 \times 8 = 17,920\,bps \approx 18\,kbps$.

S and *d* probes are TTL-limited packets. Their TTL field in IP header is set to the hop number of their probing measurable point. In addition, for d probes we also use so-called "targeted" packets. It is a UDP or an ICMP echo packet with *destination IP address* set to corresponding measurable point on the forward path. Targeted d probe has the property that its *return path* is guaranteed to share (the latter half of) the forward path, *i.e.*, the path that f probe travels. However, its *outgoing path* is not necessarily consistent with (the first half of) the backward path, *i.e.*, the path that b probe travels. On the contrary, the regular TTL-limited d probe does right the opposite thing: its outgoing path is ensured to share the backward path, while its return path is not always consistent with the forward path. Therefore, targeted and TTL-limited d probes "play well" in cases of different path patterns, thereby supplementing each other.

2) **Matching Probing Time***:* For coordinate packet probing to function effectively, it is required that probing packets travel shared congested paths nearly synchronously. In particular, we aim to achieve the following (*e.g.*, Figure 1): $send\_time(f) = send\_time(s)$, $send\_time(d) = send\_time(b)$, $recv\_time(f) = recv\_time(d)$, and $recv\_time(b) = recv\_time(s)$. In practice, we can only directly control sending times; therefore, for each measurable pair, we apply the following formulas:

$$\begin{cases} send\_time(s)\ is\ predetermined^{4} \\ send\_time(f) = send\_time(s) \\ send\_time(d) = send\_time(s) + f_{min} - d_{min} \\ send\_time(b) = send\_time(d) \end{cases} \tag{12}$$

While we do *not* require "hard" synchronization between a sender and a receiver, (provided that we are interested in measuring *relative* one-way delays) some level of synchronization is certainly needed. The key problem is to implement the third line of Equation (12), *i.e.*, match sending times of probes generated by different machines. To solve the problem, we apply a negative-feedback mechanism we explain below.

The difference between the scheduled receiving times of f and d probes is $(send\_time(s) + f_{min}) - (send\_time(d) + d_{min})$. This difference is calculated at sender and used as the negative feedback to the receiver. If the absolute value of this difference is less than 1 ms, we regard that sender and receiver have matched their probing times; otherwise, they adjust the time offset by communicating via a control channel. Upon receiving an adjust-time-offset command from the channel, the receiver re-tunes the sending times of d and b probes accordingly.

C. Adaptive Tuning

[4]Sending times of *s* probes are predetermined by the sequential scheduling algorithm (Section II-B.1).

1) **Quality of Measurability***:* A common prerequisite for *4-p*, *fsd*, and *fsb* probing techniques to work is a certain network condition (identified by Equation (2), (3), or (5)). Such conditions are not strong, but must be effectively checked. To facilitate this checking we define *quality of measurability* (QoM) for each of the three techniques as follows:

$$\begin{cases} QoM_{4P} & = 1 - \frac{|(\Delta f+\Delta b)-(\Delta s+\Delta d)|}{max(\Delta f+\Delta b,\ \Delta s+\Delta d)} \\ QoM_{fsd} & = 1 - \frac{|\Delta f-(\Delta s+\Delta d)|}{max(\Delta f,\ \Delta s+\Delta d)} \\ QoM_{fsb} & = 1 - \frac{|\Delta s-(\Delta f+\Delta b)|}{max(\Delta s,\ \Delta f+\Delta b)}. \end{cases} \quad (13)$$

A QoM quantifies the extent to which a given condition is satisfied. If it equals 1, the condition is fully met; the closer to zero it is, the poorer the condition. Also, as inferred from Section II-A.5, no QoM is associated with *2-p* probing, since it is guaranteed that f and s probes share the first half of the forward path.

From each probe set sent to a given measurable point, we calculate an instantaneous *QoM* value. However, a single instantaneous *QoM* sample is insufficient to accurately determine whether a certain probing technique is suitable for a given measurable point. Thus, we use an *exponentially weighted moving average* to determine the QoM trend over longer (than several seconds) time scales. Comparing the moving average of QoM with certain thresholds, we decide whether we should promote to or demote from certain probing techniques. The thresholds settings are shown in Table III. They are tuned based on Internet experiments. Moreover,

Technique	Promote to the technique	Demote from the technique
4-p probing	$QoM_{4P} > 0.6$	$QoM_{4P} < 0.5$
fsd probing	$QoM_{fsd} > 0.6$	$QoM_{fsd} < 0.5$
fsb probing	$QoM_{fsb} > 0.5$	$QoM_{fsb} < 0.4$

TABLE III
QOM THRESHOLD SETTINGS

we compute confidence for each individual QoM value. The larger the queuing delays observed by a probe set, the larger the corresponding confidence value, and the larger the weight given to this sample when computing the moving average. Finally, to tell whether a measured queuing delay (*i.e.*, Δf, Δs, Δd, and Δb) is large or not, we set an appropriate threshold by using an algorithm we explain in Section III-E.1 below.

2) **Technique Hopping and Adaptive Pairing***:* Here, we describe how to choose an appropriate probing technique for each measurable point on an end-to-end path. The most challenging sub-problem is how to pair s and d probes for *4-p* and *fsd* probing.

During the bootstrap, we run our version of the *traceroute* program to explore hop information on an end-to-end path. Once we know the hop IP addresses in both forward and backward directions, we compare these IP addresses and make our best guess about their relationship; more specifically, which hops are joint nodes[5] of the two paths. If a hop is considered to be a joint node, we pair up the appropriate *s* and *d* probes for this hop, and set the initial technique to *4-p* probing. During the measurement, we keep track of the quality of measurability (Equation (13)) for this hop. If it is not sufficiently high, we demote the probing technique.

For all those hops on the forward path that are not considered to be joint nodes, we simply initialize the technique to *2-p* probing. However, during the measurement, we still try to *promote* the probing technique by pairing nodes on the forward path with those on the backward path. We check QoM_{4P}, QoM_{fsd}, and QoM_{fsb} for these pairings. If any of the *QoMs* become sufficiently high (*e.g.*, exponentially weighted moving average above a given threshold), we promote this hop to use the corresponding technique.

Tuning probing techniques and pairing intermediate nodes adaptively is a non-trivial task. A single s probe may have multiple complementary d probes available. Moreover, the set of *free* d probes (has not yet paired up with other *s* probes) changes over time due to technique demotions and promotions. To effectively address the problem, for each s probe, we assign a *priority* for each d probe candidate. Priority is computed based on (i) how many times we have tried pairing with this candidate, (ii) what was the average duration this candidate has "worked" from the time it is successfully paired to the time the technique is demoted, and (iii) what is the average *QoM* this candidate achieves. We give higher priority to candidates that are less frequently tried, achieve longer durations, and have larger average *QoM*.

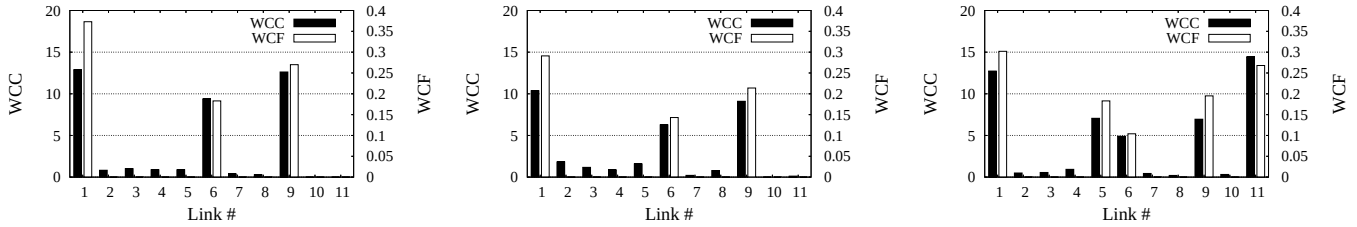
D. Evaluating Pong in Emulab

1) **Experimental Methodology***:* We evaluate *Pong*'s performance in the *Emulab* testbed. We emulate a topology consisting of 12 nodes and 11 links. Each link has a capacity of 100 Mbps and a delay of 2 ms. The two nodes at edges run *Pong* such that they measure congestion on the 11 links between them. We can generate congestion on arbitrary links. To build modest congestion on a specific link, we leverage TCP-based traffic generators provided by *Emulab*. We use periodic on/off (50% time on, 50% time off) TCP cross traffic with different periods to emulate simultaneous but independent congestion on different links. Periods are varied in the range from 0.5 to 2 seconds. In this way, we control the "ground truth" of congested links. By comparing measured congestion with the "ground truth," we check the performance of *Pong*.

2) **Experiments***:* Here, we initially evaluate the *4-p* probing technique. At first, we create congestion at links 1, 6, and 9 in the forward direction. Figure 6(a) plots *weighted congestion count* (*WCC*) and *weighted congestion frequency* (*WCF*) of each link. *WCC* indicates detected congestion on individual links via the *switch point* approach (Section II-B.2). *WCF* traces congestion status on detected congested links (Section II-C). Both *WCC* and *WCF* we show here are computed over 30-second time intervals. The figure shows that *Pong* identifies all three congested links with high accuracy.

In addition to the three strong modes at congested links' locations, *WCC* also shows some small values on other links as well. This is because the *switch point* approach assumes that probes to two neighboring nodes capture network conditions of the *same* moment. However, to improve the temporal inference

[5]We do not need to resolve aliases. A rough initial guess is enough, accurate pairing will be achieved during adaptive tuning.



(a) 3 bottlenecks originally (b) Add 2 backward bottlenecks (c) Add 2 forward bottlenecks

Fig. 6. Emulab experiment: 4-p probing scenario

characteristic of the underlying network path, we send these two probes not exactly at the same time. As a result, such inconsistency could lead to a false conclusion about congestion on an un-congested link. However, as we can see in the figure, the effect of such a false alarm is very limited, and it can easily be filtered. This is exactly the reason why *WCF* always remains zero on un-congested links.

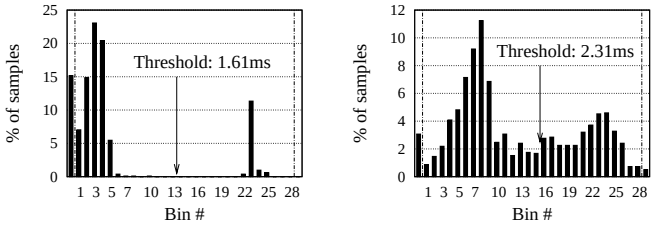
Next, we add two additional congested links (links 2 and 5) in the *backward* direction. Figure 6(b) plots measurement results in this case. As we can see, although both *WCC* and *WCF* on the three forward bottlenecks are blurred a bit, *Pong* still identifies *forward bottlenecks* with high accuracy. Finally, we remove these two backward bottlenecks and add two more forward bottlenecks (links 5 and 11) instead. Figure 6(c) shows that *Pong* successfully identifies all the five bottlenecks.

3) **Accuracy of Probing Techniques***:* Here, we evaluate the performance of different probing techniques used by *Pong*. Our results are summarized as follows: (i) *4-p* probing gives very high resolution and accuracy in identifying multiple forward congestion points and decoupling backward congestion points. (ii) *Fsd* and *fsb* probing also show relatively high accuracy (much higher than *2-p* probing). (iii) *2-p* probing is unable to decouple backward congestion points due to using only *f* and *s* probes. However, even in the worst case scenario, where *2-p* probing is used for all intermediate nodes, a single congested link on the forward path still can be identified with high accuracy. (iv) Our adaptive tuning algorithm is sensitive to path condition changes (*e.g.*, location changes of bottlenecks, route changes of probing packets). It converges to a suitable probing technique quickly, within 5 to 10 seconds in our experiments.

E. Tuning Pong in the Internet

1) **Evolving Threshold Setting Algorithm***:* In practice, setting appropriate thresholds for queuing delays (to decide whether a path or a path segment is congested, Section II-A.7 and III-C.1) is a key to the measurement accuracy of *Pong*. However, the round-trip queuing delay, the end-to-end queuing delay, or the half-path queuing delay we can measure directly or indirectly is the *delay-sum* of many links. Due to link heterogeneity of the real Internet, setting thresholds for queuing delays is a non-trivial task. The threshold setting algorithm and queuing delay distributions we explain below are based on testbed and Internet measurements from *Emulab* and *PlanetLab* experiments.

To stat distributions, we divide the range of queuing delays into 30 bins. Bin 0 and bin 29 are two special ones. Bin



(a) An Emulab path (b) An Internet path

Fig. 7. Example queuing-delay distributions

0 means queuing delay is less than *min_value* (0.15 ms) and bin 29 means queuing delay is larger than *max_value* (24.7 ms). Bin 1 to bin 28 are regular bins, which span the range [*min_value*, *max_value*]. Bin size of bin 1 to bin 28 increases exponentially with a factor of 1.2, *i.e.*, bin 1 is [0.15 ms, 0.18 ms], bin 2 is [0.18 ms, 0.216 ms], ..., bin 28 is [20.6 ms, 24.7 ms]. Based on such bin settings, we categorize queuing delay distributions measured in the Internet into several patterns and design a threshold setting heuristic for each of these patterns.

Figure 7 shows two examples of a dominant queuing delay distribution we observe in our experiments. One is the end-to-end queuing delay distribution of the 11-link path in our Emulab experiment. The other is the end-to-end queuing delay distribution of an Internet path which also consists of 11 links. As we can see, the distribution on the Emulab path shows a sheer two modes pattern — one corresponds to samples observing no congestion, the other corresponds to samples observing congestion. While the distribution on the Internet path is inherently blurred, we can still observe two relatively strong modes. In both scenarios, we set the threshold to a proper value in between the two modes, as shown in the figure.

2) **Other Tunings***:*

Handling clock skew and clock jumps. Clock skew and clock jumps on end-hosts can significantly affect the accuracy of queuing delays measured by *f* and *b* probes. *Pong* detects clock skew and clock jumps by tracking changes of f_{min} and b_{min}. For example, a linear decrease of f_{min} or b_{min} in time indicates clock skew. Once clock skew or clock jump is detected, *Pong* automatically filter out affected measurement data to minimize measurement errors.

Handling route alterations. A path may experience route alterations which can cause erroneous packet-delay estimates. *Pong* can easily detect route alterations via *s* probes. Once route alteration is detected, *Pong* pauses the measurement and keeps track of route alterations. *Pong* resumes measurement when the path is converged again.

Filtering out other anomalies. *QoM* is a sensitive indicator

of errors. By using instantaneous *QoM*, we are able to filter out many other anomalies, for example, *ICMP queuing*. The time it takes a router to generate an ICMP response is usually stable and small ([15], [3]); however, this is not always the case, *e.g.*, $100 \sim 300\,ms$ periodic jumps can occur every 60 seconds as found in [3]. When anomalies happen, *QoM* shows a small instantaneous value, which in turn shapes the weight of an affected sample to a small value. Therefore, probing results affected by anomalies have extremely small effect when updating the final measurement result.

IV. Measurements

A. Experimental Setup

We perform a large-scale Internet experiment using over 300 PlanetLab hosts. Although some argue that PlanetLab does not provide a very representative view of the Internet (*e.g.*, because about 85% of the hosts are located within *GREN — Global Research and Educational Network*), it is still a very useful platform for experiments on WAN. In any case, the question whether the results from our study could be generalized to the Internet as a whole, while extremely interesting, is beyond the scope of our work here. Currently, we are unable to make any such generalizations.

Our experiment has measured 21,861 paths within 10 days. We measure each path for 100 minutes. Out of these paths, we filter the data suspected to have non-negligible measurement errors. (One major cause of such measurement errors is the undesirable environment at PlanetLab hosts, *e.g.*, quite a number of hosts are heavily loaded and clock synchronization fails.) This leaves us with 17,587 paths (80% of the original data), which is the data set we use to generate the measurement results in this section.

1) **Congestion Spikes***:* To study properties of congestion, we introduce *congestion spikes*. A congestion spike is a continuous congestion period on a path segment during which queuing delay on a given link is consistently above a threshold.

A congestion spike starts when *non-trivial* congestion is detected on the path segment and ends when no congestion, or only trivial congestion, is detected within a given period of time. In our implementation, congestion spikes on a path segment are identified by exploiting *weighted congestion frequency* (*WCF*), defined in Section II-C. It is a measure reporting the fraction of samples that observe congestion at a path segment during 30-second time intervals. A congestion spike starts when *WCF* exceeds 0.02, and ends when it falls back below this value.

2) **Intra-AS vs. Inter-AS and Edge vs. Core***:* In addition to being capable of detecting congestion locations relative to their hop distance from the endpoints, our goal is to separate congestion events happening on intra- and inter-Autonomous System (AS) links. Inter-AS links are believed to be more congested, due to complex ISP peering policies [16].

To this end, we first resolve the AS number (ASN) for each hop on a path, based on its IP address and by using the latest BGP routing tables collected by the *Route Views* project [17]. Then, we reveal the link type based on ASNs. Starting from the first hop, we move forward. If a hop shows an ASN different from the previous hop, we infer that either the link between these two hops, or the link in front of the previous hop, is the inter-AS link. However, we do not know which of the two is the actual inter-AS link. In such a case, we simply mark both links as inter-AS links. Although this results in overestimated number of inter-AS links, we find it has little affect on our final results.

In addition, we divide links into edge and core links based on their location on the path. To represent location, we use a normalized location in the format of a percent number, *i.e.*, 0% means the first link on the path, closest to the source, and 100% means the last link on the path, closest to the destination. Edge links are those have: $location < 20\%\ or\ location > 80\%$; and core links are those have: $20\% \leq location \leq 80\%$. (The value 20% and 80% are set based on the spatial distributions of congestion properties we measured. Most congestion properties show sharp changes at these two locations.) Combining the two taxonomies, we now have four types of links as shown in Table IV:

intra-AS edge link	both an edge link and an intra-AS link
inter-AS edge link	an access link of an edge AS
intra-AS core link	both a core link and an intra-AS link
inter-AS core link	both a core link and an inter-AS link

TABLE IV
Link Classification

B. Self-consistency Validation

Here, we correlate observations from different endpoints that measure the shared path segments concurrently, and determine if the measurements are consistent in both spatial and temporal domains. Consistency over a large number (*i.e.*, over 3,500 in our case) of independent measurements is a strong indicator of the validity of the approach.

We collect data useful for self-consistency validation in the following way. First, from the raw data, we collect a total of 346,591 congestion spikes. Then, we select the congestion samples that reside exclusively on *physical* links for which both ends are measurable points. In this way, we guarantee that we know the accurate location (*i.e.*, on the granularity of a single link) of these samples. This leaves us with 335,391 remaining samples.

We further reduce the data set by filtering out congestion samples that happen at network edges. We do this for two reasons. First, congestion scale on edge links is much larger than that in the core (see Section IV-D). Thus, to make our result more convincing, we prefer to show self-consistency results on links for which the congestion scale is relatively small. Second, links near the core are more likely to be measured by endpoints from different physical locations. Therefore, data on these links is more suitable for self-consistency validation.

After filtering out congestion samples residing at edges, 160,813 samples remain. These congestion samples happen on 6,168 different links and 8,552 different paths. We find 3,500 links are shared by more than one path (799 links by more than 10 paths and 60 links by more than 100 paths). Among them, we pick up the eight links that are shared by the most paths. Table V describes information of these eight

links. For example, link 2 is shared by 301 different paths; these paths source from 57 PlanetLab hosts and are destined to 35 PlanetLab hosts.

Link	Type	Path	Sender	Receiver
Link 1	inter-AS	312	3	140
Link 2	inter-AS	301	57	35
Link 3	intra-AS	295	19	69
Link 4	inter-AS	294	32	38
Link 5	inter-AS	262	120	3
Link 6	intra-AS	252	54	31
Link 7	intra-AS	243	6	110
Link 8	inter-AS	239	14	101

TABLE V
TOP EIGHT CONGESTED LINKS OBSERVED BY MOST PATHS

For each of the eight links, we check related paths by examining sets of link congestion status graphs generated by *Pong*. We find that *Pong* locates congestion points with very high accuracy. There are almost zero false positives on the links close to the shared link. For the vast majority of these paths, *Pong* shows *no* congestion on links right in front and right behind the shared link when the shared link is congested. This means that *Pong* experiences no "leaking" effects in which congestion at one location is incorrectly allocated to neighboring links. For those paths that do show congestion on neighboring links, we find that such neighboring links are the ones that are on the side closer to the edge. Such links are also congested at other moments when the shared links are not. Therefore, we conclude that such neighboring links are actually experiencing congestion instead of showing false positives caused by congestion on the shared link.

We also check temporal consistency for the congestion measured on shared links. To accomplish this, we find concurrent measurements, *i.e.*, overlapping measured time periods for paths associated with a shared link. Then, we plot and manually examine these periods. The results show relatively high temporal consistency: during an overlapping period, if on one path we observe congestion of the shared link, we can also observe it on other paths — at the same time and with similar congestion scales and time patterns.

C. Technique Utilization

Here, we summarize the results for the use of different probing techniques and show their spatial distribution. Such information reveals valuable information about underlying path topologies and spatial congestion distributions. The overall result is as follows. The *4-p* and *2-p* probing techniques are utilized 56% and 36% of time respectively on average. The *fsb* probing is utilized 7% of time, and *fsd* probing is only utilized 1% of time approximately. The large percent for *4-* and *2-p* probing is not a surprise. As explained in Section III-C.2, *4-p* probing is always tried first, and *2-p* probing is the last resort. *Fsd* and *fsb* techniques are applied only as transient techniques between the default ones. Therefore, their percent is not high. Particularly, for a large majority of cases where *fsd* probing is applicable, we can also find an appropriate pairing to make *4-p* probing work. Hence, *4-p* probing is finally used. This is exactly why percent of *fsd* probing is the lowest.

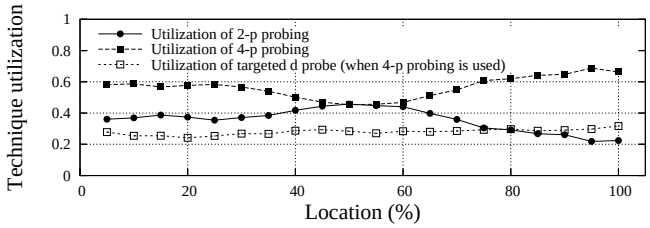


Fig. 8. Technique utilization vs. location

Figure 8 shows spatial locations for *4-p* and *2-p* measurable points as well as for routers supporting targeted d probes. Figure 8 shows that *4-p* and *2-p* measurable points prevail: their sum is typically above 90%. However, the spatial distribution is not uniform. For the links closer to edges the *4-p* technique prevails, which means there are many complementary *d* probes available. For the links closer to the core (*e.g.*, 50% location), the amount of complementary d probes decreases. This is because forward and backward paths are typically disjoint at the core [10]. However, the figure shows that many of the intermediate nodes on *disjoint* paths in the core (over 45%) could still be successfully paired.

D. Spatio-temporal Congestion Properties

To characterize Internet congestion properties in a fine-grained and diversified manner, we introduce novel measures. Below, we present these measures (Table VI) and summarize their average values (Table VII). At the beginning of each forthcoming subsection, we explain each measure and discuss its spatio-temporal properties.

Measure	Meaning
CD	*Congestion duration*; duration of a congestion spike.
CI	*Congestion intensity* of a congestion spike.
SCT	*Segment congestion time*.
SCF	*Segment congestion frequency*.

TABLE VI
SUMMARY OF CONGESTION MEASURES

	*CD**	*CI*	*SCT*	*SCF*	$\frac{SCT}{SCF}$
All	1.93	0.063	0.0033	0.032	0.105
Intra-AS	2.08	0.070	0.0046	0.041	0.113
Inter-AS	1.54	0.043	0.0011	0.019	0.059
Edge	2.31	0.081	0.0058	0.047	0.124
Core	1.50	0.044	0.0013	0.020	0.068
Intra-AS (edge)	2.37	0.084	0.0068	0.054	0.124
Inter-AS (edge)	1.53	0.044	0.0011	0.018	0.060
Intra-AS (core)	1.50	0.045	0.0016	0.021	0.074
Inter-AS (core)	1.49	0.043	0.0011	0.019	0.057

* Values of *CD* are in minutes.

TABLE VII
AVERAGE VALUES OF CONGESTION MEASURES

1) **Congestion Duration***:* We define a congestion spike as a continuous congestion period on a path segment (Section IV-A.1). *Congestion duration* is simply the spike duration. Our measurements over all collected spikes indicate that the spike duration is not small. Its average is about 2 minutes, while 10% of the spikes last longer than 4 minutes.

Edge vs. core (shown in Table VII (column *CD*)). Averaging over all congestion spikes, congestion duration at edges is larger than that at the core (2.31/1.5; 1.54 times larger). The pattern is almost the same (2.37/1.5; 1.58 times larger) if we

only consider congestion spikes on intra-AS links. However, if we only consider congestion spikes on inter-AS links, it does not show this pattern. On inter-AS links, congestion duration at edges is very similar to that at the core (1.53/1.49). Nevertheless, as we can see, the overall pattern is dominated by the pattern on intra-AS links.

Inter-AS vs. intra-AS (shown in Table VII (column *CD*)). Averaging over all congestion spikes, congestion duration on intra-AS links is larger than that on inter-AS links (2.08/1.54; 1.35 times larger). If we only consider congestion spikes at edges, congestion duration on intra-AS links is 1.55 (2.37/1.53) times larger than that on inter-AS links, larger than the overall average. However, if we only consider congestion spikes at the core, congestion duration on intra-AS links becomes almost the same as on inter-AS links (1.50/1.49).

2) **Congestion Intensity***: Congestion intensity* of a congestion spike is defined as the averaged *weighted congestion frequency* (*WCF*) during the congestion spike. Below, we focus on the spatial-temporal distribution of this measure.

Edge vs. core (shown in Table VII (column *CI*) and Figure 9(a)). Figure 9(a) shows that the congestion intensity at edges is larger than that at the core. For example, 15% of spikes at edges have congestion intensity larger than 0.1, while only 4% of spikes in the core achieve such intensity. If we only consider congestion spikes on intra-AS links, this pattern becomes even more obvious (not shown in the figure; the result is available in Table VII). Also, if we only consider congestion spikes on inter-AS links, the congestion intensity at edges is very similar to that at the core.

Inter-AS vs. intra-AS (shown in Table VII and Figures 9(b), 9(c), 9(d)). Figure 9(b) shows that congestion intensity on intra-AS links is larger than that on inter-AS links. For example, 12% of congestion spikes on intra-AS links are "stronger" than 0.1; at the same time, only approximately 3% of the spikes on inter-AS links have their intensity larger than 0.1. If we only consider the edge (Figure 9(c)), the inter- vs. intra-AS intensity increases – CDF curves are more distant from each other. However, if we only consider the core (Figure 9(d)), congestion intensities on inter-AS and intra-AS links are almost the same.

3) **Segment Congestion Time***:* For a given path segment, we define the normalized *segment congestion time* as the ratio between the number of probe samples reporting congestion on that segment and the total number of probe samples available for that segment during the entire measurement. Simply speaking, it indicates the fraction of time the segment is experiencing congestion.

Figure 10(a) depicts the complementary CDF (CCDF) characteristic for the segment congestion time for all measured segments (the curve is marked by "all"). The curve shows that 17% path segments experience non-trivial congestion (congestion time larger than 0.001); 5% path segments have the normalized congestion time larger than 0.01, and 1% path segments have the congestion time larger than 0.1 (*i.e.*, they are congested more than 10% of the total measured time). Our experiment was run *continuously* over 10 days. As such, our results include not only day-time measurements, but also nearly zero-congestion night-time periods.

Edge vs. core (shown in Table VII (column *SCT*) and Figure 10(a)). Averaging over all measured path segments, segment congestion time at edges is *4.5 times* larger than that at the core (0.0058/0.0013). If we only consider intra-AS links, the above ratio remains almost the same (0.0068/0.0016). However, if we only consider inter-AS links, there is no such pattern. As we can see, segment congestion times at edges and in the core become the same (both are 0.0011).

Inter-AS vs. intra-AS (shown in Table VII (column *SCT*) and Figures 10(b), 10(c), 10(d)). The segment congestion time statistics shows similar inter- vs. intra-AS patterns as the above measures.

4) **Segment Congestion Frequency***:* For a given segment, we define the *segment congestion frequency* (shown in Table VII (column *SCF*)) as the ratio between the number of 30-second time periods during which non-trivial congestion is observed on that segment, and the total number of 30-second periods during the entire measurement. This metric represents the congestion frequency over a longer time scale.

The ratio between the segment congestion frequency and the normalized congestion time, $\frac{SCT}{SCF}$, describes the overall "congestion density" ($\in [0, 1]$). The larger the value, the more clustered in time the congestion events are. Our experiment indicates that the congestion density (shown in Table VII, column $\frac{SCT}{SCF}$) monotonically decrease from edges to the core (*e.g.*, 0.124 vs. 0.068). This reveals that congestion events at edges are relatively clustered in time, while dispersed at the core.

E. Path-level Congestion Properties

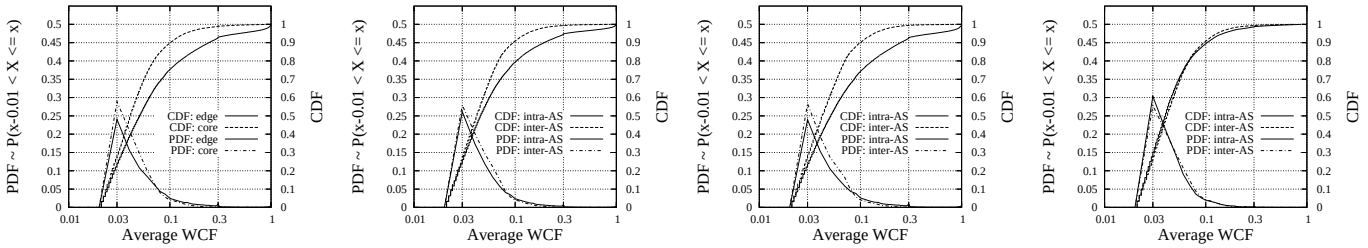
1) **Path Congestion Time***:* To characterize the path-level congestion, we simply sum up the normalized segment congestion times for all segments that constitute this path, and obtain the normalized *path congestion time*. If the path congestion time is less than 0.002, the path is considered to have *no or trivial* congestion. Otherwise, if it is less than 0.03, we characterize the path as being *slightly* congested. If the path congestion time resides between 0.03 and 0.15, we regard this path as being *moderately* congested. Finally, if the path congestion time is larger than 0.15, we characterize the path as being *considerably* congested. Table VIII summarizes the congestion properties of measured paths.

All paths	17,587	100%
Paths with no or trivial congestion	8,434	48.0%
Paths with slight congestion	5,453	31.0%
Paths with moderate congestion	2,412	13.7%
Paths with considerable congestion	1,288	7.3%

TABLE VIII
PATH-LEVEL CONGESTION PROPERTIES

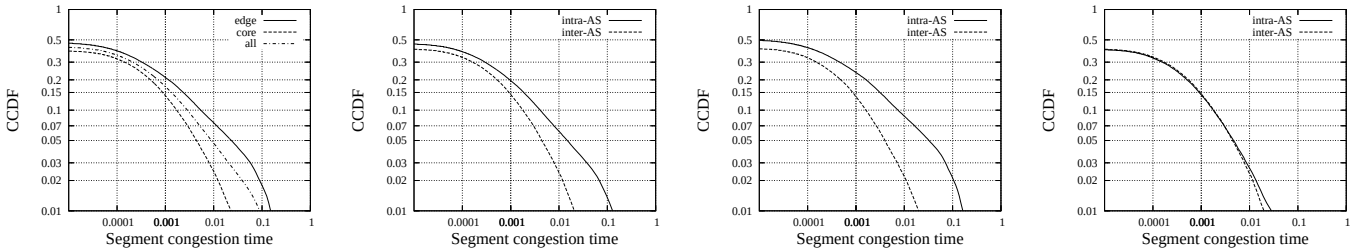
The result shows that approximately 52% of paths experience non-trivial congestion. Thus, despite relatively moderate segment congestion time (Section IV-D.3), the path congestion time is considerable due to an accumulation effect — an end-to-end path "picks up" congestion on all underlying links.

2) **Multiple Congestion Points***:* Figure 11 shows the probability of observing one or more congested points on an end-to-end path as a function of time-scales. Naturally,



(a) Edge vs. core (b) *Intra-AS* vs. *inter-AS* (c) At edges (d) At the core

Fig. 9. PDF and CDF of congestion intensity



(a) All vs. edge vs. core (b) *Intra-AS* vs. *inter-AS* (c) At edges (d) At the core

Fig. 10. CCDF of segment congestion time

over longer time-scales, the probability to observe congestion increases. For example, the probability to observe at least one congested point over 30-second-long time intervals is 0.25, while it becomes as much as 0.46 for 30-minute-long intervals. Likewise, for a given time-scale, the probability to observe multiple congestion points decreases. For example, for 1.5-minute-long time intervals, the probability to observe two or more congested points is 0.1. Yet, the probability to see 10 or more congested points within the same interval is below 0.001.

With nonlinear least squares regression, we find that the probability P to observe more than N congestion points simultaneously over T minutes long time intervals can be approximated by the following relationship:

$$P \propto \frac{T^{\alpha}}{e^{\beta N}} \quad (\alpha = 0.4 \sim 0.5,\ \beta = 0.4 \sim 0.6) \qquad (14)$$

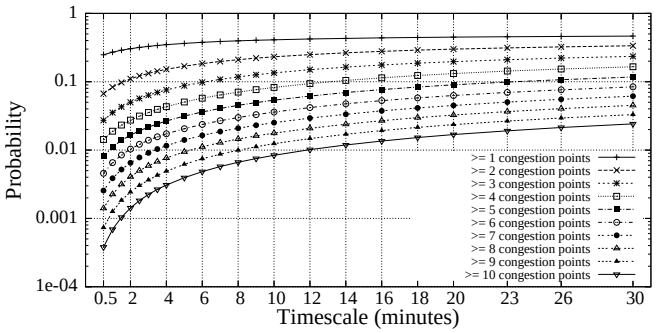


Fig. 11. Probability of multiple congestion points

V. Related Work

Active measurements can be divided into two categories: end-to-end approaches (*e.g.*, [18], [12], [13], [2], [14], [19]) and router-response-based approaches (*e.g.*, [20], [21], [3]). End-to-end approaches send single probe packets, packet pairs, or packet trains. They measure one-way delay or packet dispersion and infer end-to-end path properties such as bottleneck capacity ([12], [22], [19]), available bandwidth ([18], [14], [23]), or bottleneck location ([2], [23]). Router-response-based approaches probe intermediate routers along a path and diagnose link-level properties such as queuing delay, packet reordering, packet loss rate, and the corresponding location.

Pong combines approaches of both categories and focuses on inferring link-level queuing delays and congestion locations. *Pong* sends both end-to-end and router-targeted probes and correlate them in a novel way. Unlike end-to-end tools such as *Pathneck* [2] or *Stab* [23], which can only locate the dominant bottleneck or minor bottlenecks in front of the dominant one, *Pong* is able to locate all congested points on a path. Unlike router-response-based tools such as *Tulip* [3] or *Pathchar* [21], *Pong* correlates probes sent to different intermediate routers in a more effective way, thereby achieving a higher accuracy with a much lower traffic overhead.

BFind [24] exploits TCP's property of gradually filling up the available bandwidth and combines it with router-targeted probes to measure the location of dominant bottlenecks. Their experiments on PlanetLab show that *non-access* bottlenecks are equally split between intra-ISP links and peering links between ISPs. This is consistent with our results (Section IV-D) that congestion beyond the edge behaves similarly on intra-AS links and inter-AS links. As recognized by its authors, the main drawback with *BFind* is that it is a heavy-weight tool that sends a large amount of data. As a result, it is only applicable for short duration measurements. In comparison, the overhead of *Pong* is very low, hence *Pong* is also competent for continuous monitoring.

Gurewitz and Sidi perform a mathematical study on estimating one-way delays from round-trip delay measurements [25]. However, their approach is difficult to apply in practice for two reasons: (i) it requires knowledge of the exact path traveled by a round-trip probe, while in practice the returning path from a router usually can not be measured; (ii) even if exact paths can be measured, the set of paths still has a high probability not to satisfy the topology relations required by their algorithm. In contrast, since *Pong* measures queuing delays instead of absolute delays, it can easily mitigate the

two problems — *Pong* tolerates agnostic of paths that probes actually travel; also, the desirable conditions for *Pong* to work have a high probability to be satisfied in practice (Section II-A.2). In addition, the approach of [25] requires measurements from a large number of vantage points, while *Pong* only needs measurements from the source and the destination.

Network tomography approaches (*e.g.*, [26], [27], [28], [29], [30], [31]) exploit temporal correlations among multiple receivers in a multicast-like environment and infer link level characteristics such as delay and loss by only using end-to-end probing. A recent work, *Mils* [9], has developed an unbiased algebraic model that significantly reduces the complexity of network tomography and improves its inference accuracy. However, in comparison with *Pong*, even when equipped with *Mils*, a network tomography approach is still more complex. In addition, since network tomography infers link properties by measuring *shared* segments of multiple paths, it does not work if an objective link can not be mapped to a shared segment. Therefore, its measurable range on an individual path is limited. On the contrary, *Pong* is able to measure all links on a path. Moreover, it is challenging to deploy a network tomography tool to generate measurement results in real time as *Pong* does.

The *network radar* approach ([32], [33]) uses only round-trip time measurements, and thus no longer requires the collaboration from receivers. As a result, it makes the measurements much easier to deploy and significantly expands the scope of measurable ranges on individual paths. However, the tradeoff is radar's vulnerability to interferences from shared segments on return paths and from "noises" caused by receiving hosts, thereby reducing its measurement accuracy and robustness. In contrast, *Pong* achieves both high accuracy and robustness in practice. In addition, although *network radar* significantly expands the measurable ranges on individual paths in comparison with traditional network-tomography approaches, it is still unable to measure an *arbitrary* link on a path as *Pong* does.

End-to-end approaches have also been used to detect shared congestion of multiple paths. For example, Rubenstein *et al.* [34] detect shared points of congestion by taking advantage of Poisson probes. Similarly, Kim *et al.* [35], [36] detect shared congestion by using a wavelet denoising approach. Because both approaches strictly leverage end-to-end measurements, they must rely on advanced probing patterns and signal processing methods to solve the problem. On the contrary, *Pong* combines end-to-end with router-response-based probes, and provides an effective way to directly measure congestion properties on individual links. As such, it can lead to much more effective methods to measure shared congestion. We leave this for our future work.

VI. Conclusions

In this paper, we presented *Pong*, a measurement tool we designed to accurately reveal spatio-temporal Internet congestion properties. *Pong* can be used for long term monitoring in addition to on-demand measurement due to its lightweight nature. In our approach, endpoints (i) actively probe the network in a coordinated manner to estimate path congestion properties; (ii) strategically combine end-to-end and router-based probes to achieve high sampling frequency; (iii) effectively self-adapt to the inferred topological characteristics and path congestion properties to improve the quality of measurement; and (iv) accurately detect and filter out periods of measurement inaccuracy. Experiments in the Emulab testbed verify that *Pong* achieves high accuracy in both spatial and temporal domains.

We deployed *Pong* in PlanetLab, and measured spatio-temporal characteristics of over 21,000 Internet paths. Our findings are as follows. (i) The network edge is more frequently congested than the core; *4.5 times* on average. (ii) However, once the congestion happens, the congestion intensity at the edge is on average only 1.84 times larger than that in the core. (iii) Overall, intra-AS links are more congested than inter-AS links. More precisely, most congestion occurs on *intra-AS links at edges*. (iv) At the Internet edges, intra-AS links are much more congested than inter-AS ones. (v) In the core, congestion on intra-AS links and congestion on inter-AS links are similar. (vi) Intra-AS links at the edge are significantly more congested than intra-AS links in the core. (vii) For inter-AS links, there is little to no difference in the congestion at the edge and in the core. ($viii$) Congestion events at edges are relatively clustered in time, while dispersed in the core. (ix) Approximately 17% of segments we measured experience congestion; but only 1% of segments experience congestion more than 10% of time. (x) Almost 52% of end-to-end paths experience non-trivial congestion and 7.3% of them experience considerable congestion. (xi) The probability to observe multiple congested points on an end-to-end path over a given time interval grows as a power function of the interval length, and decays exponentially with the number of congested points.

References

[1] L. Deng and A. Kuzmanovic, "Pong: Diagnosing spatio-temporal internet congestion properties," in *ACM SIGMETRICS (poster)*, San Diego, CA, June 2007.

[2] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang, "Locating Internet bottlenecks: Algorithms, measurements, and implications," in *ACM SIGCOMM*, Portland, Oregon, Sept. 2004.

[3] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level Internet path diagnostics," in *ACM SOSP*, Bolton Landing, NY, Oct. 2003.

[4] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the consistency of Internet path properties," in *ACM SIGCOMM IMW*, San Francisco, CA, Nov. 2001.

[5] A. Kuzmanovic and E. Knightly, "TCP-LP: Low-priority service via end-point congestion control," *IEEE/ACM ToN*, vol. 14, no. 5, Oct. 2006.

[6] D. Wei, C. Jin, S. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM ToN*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006.

[7] A. Jain, J. M. Hellerstein, S. Ratnasamy, and D. Wetherall, "A wakeup call for internet monitoring systems: The case for distributed triggers," in *HotNets*, San Diego, CA, Nov. 2004.

[8] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "PlanetSeer: Internet path failure monitoring and characterization in wide-area services," in *OSDI*, San Francisco, CA, December 2004.

[9] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, Pisa, Italy, Sept. 2006.

[10] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM ToN*, vol. 5, no. 5, pp. 601–615, Oct. 1997.

[11] "PlanetLab," http://www.planet-lab.org/.

[12] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[13] ——, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM ToN*, vol. 12, no. 6, pp. 963–977, 2004.

[14] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[15] R. Govindan and V. Paxson, "Estimating router ICMP generation delays," in *PAM Workshop*, 2002.

[16] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the causes of path inflation," in *ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[17] "University of Oregon Route Views Archive Project," http://www.routeviews.org/.

[18] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan, "PCP: Efficient endpoint congestion control," in *NSDI*, San Jose, CA, May 2006.

[19] S. Saroiu, P. Gummadi, and S. Gribble, "Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments," in *IEEE INFOCOM*, New York, NY, June 2002.

[20] K. Anagnostakis, M. Greenwald, and R. Ryger, "cing: Measuring network-internal delays using only existing infrastructure," in *IEEE INFOCOM*, San Francisco, CA, June 2003.

[21] A. Downey, "Using Pathchar to estimate Internet link characteristics," in *ACM SIGCOMM*, New York, NY, Oct. 1999.

[22] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: a simple and accurate capacity estimation technique," in *ACM SIGCOMM*, New York, NY, 2004.

[23] V. Ribeiro, R. Riedi, and R. Baraniuk, "Locating available bandwidth bottlenecks," *IEEE Internet Computing*, vol. 8, no. 5, Sept. 2004.

[24] A. Akella, S. Seshan, and A. Shaikh, "An Empirical Evaluation of Wide-Area Internet Bottlenecks," in *ACM SIGMETRICS*, San Diego, CA, June 2003.

[25] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[26] A. Adams et al, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications*, May 2000.

[27] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, New York, NY, 2002.

[28] N. Duffield, "Simple network performance tomography," in *ACM IMC*, New York, NY, 2003.

[29] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of Internet link lossiness," in *IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[30] F. Presti, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal delay distributions," *IEEE/ACM ToN*, vol. 10, no. 6, pp. 761–775, 2002.

[31] Y. Tsang, M. Coates, and R. Nowak, "Network delay tomography," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2125–2136, Aug. 2003.

[32] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "Network radar: tomography from round trip time measurements," in *ACM IMC*, New York, NY, 2004.

[33] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak., "On the performance of round trip time network tomography," in *IEEE ICC*, June 2006.

[34] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM ToN*, vol. 10, no. 3, pp. 381–395, 2002.

[35] M. Kim, T. Kim, Y. Shin, S. Lam, and E. Powers, "A wavelet-based approach to detect shared congestion," in *ACM SIGCOMM*, Portland, Oregon, Sept. 2004.

[36] ——, "Scalable clustering of Internet paths by shared congestion," in *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

Leiwen Deng is a Ph.D. student in the Department of Electrical Engineering and Computer Science at Northwestern University. He received his B.S. degree in communication engineering from Shanghai Jiaotong University, China in 1998. He was conferred the CCIE (Cisco Certified Internetworking Expert) certificate in 2001. His research interests are in the area of computer networking with emphasis on the sustainable development of Internet.

Aleksandar Kuzmanovic is an assistant professor in the Department of Electrical Engineering and Computer Science at Northwestern University. He received his B.S. and M.S. degrees from the University of Belgrade, Serbia, in 1996 and 1999 respectively. He received the Ph.D. degree from the Rice University in 2004. His research interests are in the area of computer networking with emphasis on design, measurements, analysis, denial-of-service resiliency, and prototype implementation of protocols and algorithms for the Internet.

APPENDIX I
DEVELOPING BOUNDS FOR HALF-PATH QUEUING DELAYS

First, we explain the method we use to estimate Δf_s in the *4-p* probing technique. For convenience, we replicate Equation (1) from Section II-A.1 here as Equation (15). This is the set of equations that hold in the *4-p* probing scenario.

$$\begin{cases} \Delta f = \Delta f_s + \Delta f_d \\ \Delta b = \Delta b_s + \Delta b_d \\ \Delta s = \Delta f_s + \Delta b_s \\ \Delta d = \Delta f_d + \Delta b_d \\ \Delta f + \Delta b = \Delta s + \Delta d \end{cases} \tag{15}$$

We start the estimation with two candidate ranges:

$$\begin{cases} Candidate\ range\ 1: \ \Delta f_s \in [\Delta f - \Delta d,\ \Delta f] \\ Candidate\ range\ 2: \ \Delta f_s \in [\Delta s - \Delta b,\ \Delta s] \end{cases} \tag{16}$$

In the ideal scenario, the lower bound of the two ranges should be the same, *i.e.*, $\Delta f - \Delta d = \Delta s - \Delta b$. However, in practice, these two values could be differed to some extent due to measurement errors caused by imperfect synchronization between probes sent from the two endpoints. Therefore, after aligning the lower bounds in a conservative way, we take an intersection of the two ranges to reach the result $[L_{fs},\ U_{fs}]$.

The method to estimate Δf_d is essentially the same, the only difference is that we start with the following candidate ranges:

$$\begin{cases} Candidate\ range\ 1: \ \Delta f_d \in [\Delta f - \Delta s,\ \Delta f] \\ Candidate\ range\ 2: \ \Delta f_d \in [\Delta d - \Delta b,\ \Delta d] \end{cases} \tag{17}$$

We can also estimate the two half-path queueing delays in the backward direction (*i.e.*, Δb_s and Δb_d) in a similar way.

As we can see, in the *4-p* probing scenario, we have two candidate ranges when estimating each half-path queuing delay. This provides certain redundancies, which is an important reason why *4-p* probing can be more accurate than all the other three probing techniques. The methods we use to estimate half-path queuing delays in *fsd*, *fsb* and *2-p* probing scenarios do not have redundant measurement data. We simply follow Equations (4), (6) and (7) respectively in Section II-A to develop bounds for the corresponding half-path queuing delays.