In Proceedings of IEEE ICNP 2008

Monitoring Persistently Congested Internet Links

Leiwen Deng and Aleksandar Kuzmanovic
Department of Electrical Engineering and Computer Science, Northwestern University
Email: {karldeng, akuzma}@cs.northwestern.edu

***Abstract*—Measurement tools that can accurately locate and monitor congested Internet links would significantly help us understand how the Internet operates. However, developing such tools is challenging, especially when our concerned target is congestion on the core Internet links rather than that on the relatively easily measured access links. Congestion on core links — and persistent congestion in particular — can reveal systematic problems such as routing pathologies, poorly-engineered network policies, or non-cooperative inter-AS relationships.**

In this paper, we present *Pong*, a novel tool capable of accurately locating and monitoring a subset of non-access Internet links that exhibit persistent congestion over longer time scales. Pong takes advantage of the persistently congested link property to overcome the long-lasting challenges common for delay-based inference tools. In addition, it exploits the same property to (i) infer otherwise unknown underlying path conditions, (ii) determine appropriate queuing delay thresholds to reveal congestion, (iii) achieve high accuracy with low probing rate, and (iv) detect moments of its own inaccuracy. Finally, Pong can quantify measurement results' accuracy comprehensively, allowing us to further select vantage points that maximize the observability of the underlying congestion.

I. Introduction

Measurement tools that can accurately locate and monitor congested Internet links would significantly help us understand how the Internet operates. While developing a tool able to locate congested links is already an ambitious goal, making it capable of monitoring congested links as well becomes even harder. In particular, when we want to perform very long term monitoring tasks, we need a lightweight tool that induces a very low traffic overhead.

In addition, congestion on non-access links is usually much harder to measure than that on access links. This is because when we perform measurements from Internet edges, the access links, which are on average much more congested, can often overshadow our congestion observation for those non-access links behind them. However, the congestion on non-access links might reflect more about underlying problems of the Internet.

In this paper, we present *Pong*, a lightweight measurement tool that can both locate congested links accurately and monitor them in the long term. The particular measurement target that Pong is optimized for is a subset of non-access links that exhibit relatively persistent congestion. This subset of congested links can reveal systematic problems of the Internet such as routing pathologies, poorly-engineered network policies, or non-cooperative inter-AS relationships.

Pong measures queuing delays as congestion indicators. It dramatically increases observability of links behind an access link by sending probes from both endpoints of a path and by combining end-to-end probes with probes to intermediate routers when measuring queuing delays. In addition, based on its continuous monitoring capability, Pong improves its accuracy in locating repetitively congested links (those that experience frequent queue building-up and draining epochs over longer time scales) using measurement statistics over a longer time period.

When measuring queuing delays from both endpoints, Pong uses a novel method to infer whether the different paths traveled by different probes share the same persistently congested links. For example, whether the end-to-end probe sent from the source endpoint travels a congested link shared with the returning path of a router-targeted probe (returned as an ICMP response) sent from the destination endpoint. It can therefore correlate the probes concurrently traveling shared congested links to infer congestion locations. The above method does not need to resolve the actual routes of these paths, but simply checks whether values of measured queuing delays on these paths satisfy certain relationships.

Pong carefully handles issues in the real Internet environment to optimize its practical measurement accuracy. It applies an adaptive algorithm to set proper queuing delay thresholds that determine congestion on a per-path basis. It detects anomalies such as clock skews and jumps at end hosts, route alterations of paths, and ICMP queuing at routers. It reacts to the anomalies by either filtering affected measurement samples or suspending the measurement until the anomalies are relieved.

Pong provides a distinct solution for scenarios when its measurement accuracy is significantly degraded due to undesirable path conditions. To this end, it quantifies its measurement accuracy for specific congested links on specific paths in a comprehensive way. As a result, it allows us to select paths that can give the best measurement accuracy for concerned links. This is particularly meaningful for measuring congestion on non-access links which can usually be observed from many paths with very different path conditions.

The reminder of this paper is organized as follows: In Section II, we introduce Pong's methodology of accurately locating and monitoring repetitively congested links. In Section III, we address measurement issues in the real Internet environment. In Section IV, we analyze Pong's performance in practice via Internet experiments. In Section V we present related work. Finally, we conclude in Section VI.

II. Methodology

In this section, we introduce our methodology for the link congestion measurement. This includes (i) how to exploit

coordinated probing from both endpoints of a path to measure the congestion location relative to each intermediate router, and (ii) how to correlate measurement results for neighboring intermediate nodes (routers) to locate congested links. We also show a measurement example that demonstrates the effectiveness of our methodology.

A. Coordinated Probing

Coordinated probing is a key for our proposed measurement methodology. It exploits coordinated active probing from both endpoints of a path and strategically combines end-to-end probes with probes to intermediate nodes. Its goal is to generate queuing delay estimates for the two *half paths* separated by a concerned intermediate router on the path. Based on the queuing delay estimates, we can tell whether each half path is congested or not.

1) **A Simplified Case — Symmetric Path:** Consider a simplified symmetric path scenario shown in Figure 1. For each intermediate node (assuming that it responds to TTL-limited probes), we send four types of probing packets from the two endpoints: (i) an *f* ("forward") probe from the source to the destination; (ii) a *b* ("backward") probe from the destination to the source; (iii) an *s* ("source") probe from the source but with its TTL expired at the concerned intermediate node, thereby returning to the source in form of an ICMP packet; (iv) a *d* ("destination") probe which is similar to the *s* probe, but is both sent from and returned to the destination.

Each of the four probing packets measures the delay of the path that it travels. Based on that we can compute the corresponding queuing delay. Denote by Δf, Δb, Δs, and Δd the queuing delays of paths measured by *f*, *b*, *s*, and *d* probes, respectively. The queuing delay is computed by subtracting the minimum delay value (within a long enough recent sample history) from a sampled delay.

Based on the four queuing delays, we can estimate queuing delays of the two half paths. Denote by Δf_s and Δf_d the queuing delays of the two half paths in front of and behind the concerned intermediate node as shown in Figure 1. Although we can not deterministically compute the values of Δf_s and Δf_d, we can effectively estimate their ranges. 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 Δd or Δb is small, we have a tight bound for Δf_s.

A prerequisite for this method to work is that we should coordinate probing time of the four types of probes such that they capture queuing delays of nearly the same moments for those shared path segments that they travel. We achieve this in the following way: (i) At the source node, we send the *f* probe and the *s* probe simultaneously. (ii) At the destination node, we send the *d* probe and the *b* probe simultaneously. (iii) Between the source and destination, we coordinate the sending time of the *f* and *d* probes such that it satisfies the following condition: the expected receiving time of the *f* and the *d* probes at the destination node should be nearly the same. The expected receiving time is computed as the sending time plus the transmission delay on a path. The transmission delay is approximated using the minimum delay measured by the *f* and *d* probes (within a long enough recent sample history).

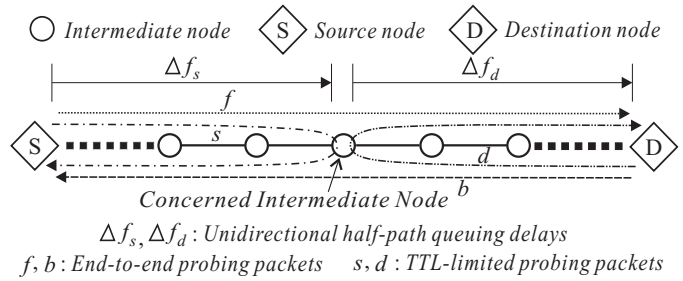


Fig. 1. Coordinated probing: a symmetric path scenario

2) **General Cases — Asymmetric Path:** Our proposed methodology works for asymmetric paths as well. In practice, it is non-trivial to check whether a path is symmetric or not. For example, one approach is to compare traceroute results of the forward and backward paths and then resolve aliases for intermediate nodes. However, to be able to use the method described in the previous section to estimate Δf_s and Δf_d, we do *not* have to resolve whether a path is exactly symmetric. Instead, we can check a more relaxed condition and do so in a much simpler way than to resolve the path symmetry. This relaxed condition corresponds to a specific *path pattern*, of which the symmetric path scenario is just a special case, as we explain in detail below.

The 4 packet (4-p) probing scenario. The measured queuing delays in the symmetric path case should satisfy the following condition (if there are no measurement errors, the "$\approx$" becomes "="):

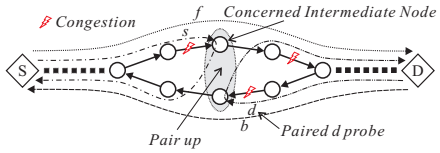
$$\Delta f + \Delta b \approx \Delta s + \Delta d \tag{1}$$

Indeed, this condition sufficiently indicates a path pattern, for which we can estimate Δf_s and Δf_d in the same way as in the symmetric path case. We denote this path pattern as the *4-p probing scenario* in the sense that we use all four types of probes to generate the estimates for Δf_s and Δf_d.

Paths in the 4-p probing scenario do not have to be symmetric. Many asymmetric paths can still satisfy Condition (1). The first column of Table I shows such an example. In this example, the *s* probe and the *d* probe are targeted to two different intermediate nodes (one is the concerned intermediate node, the other is a node on the backward path), but we can pair them up. There are three repetitively congested links in this example. One is captured by both the *f* and *s* probes, the another one is captured by both the *f* and *d* probes, and the last one is captured by both the *d* and *b* probes. Therefore, queuing delays measured by *f*, *b*, *s*, and *d* probes still satisfy Condition (1). We can then use proper formulas (highlighted in Table I) to estimate Δf_s and Δf_d for the two forward half paths separated by the concerned intermediate node.

Fsd, fsb, and 2 packet (2-p) probing scenarios. We define three different path patterns in general. Each of them indicates a specific pattern for route relationship among paths (traveled by *f*, *b*, *s*, and *d* probes) and locations of repetitively congested links on these paths. For each of them, we can use a respective set of formulas to estimate Δf_s and Δf_d. We name the three patterns after the probes that they use when estimating Δf_s and Δf_d, and they are: (i) *4-p probing scenario* mentioned

4-p probing scenario

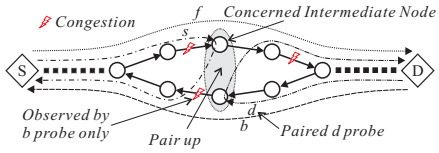


$Pattern\ Condition:\ \Delta f + \Delta b \approx \Delta s + \Delta d$

Formula highlights

(Take Δf_s for example, estimating Δf_d is similar.) Use $\Delta f_s \in [\Delta f - \Delta d, \Delta f]$ if Δd is very small. Use $\Delta f_s \in [\Delta s - \Delta b, \Delta s]$ if Δb is very small. Otherwise, use both ranges to compute a range which is a proper linear combination of the two. The linear combination takes into account measurement errors of Δf, Δd, Δs, and Δb. We estimate measurement errors based on measured deviation of recent samples.

Fsd probing scenario

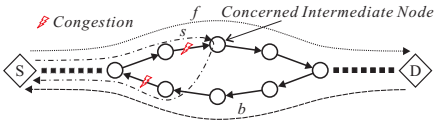


$Pattern\ Condition:\ \Delta f \approx \Delta s + \Delta d$ (2)

Formulas to estimate Δf_s and Δf_d

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

Fsb probing scenario



$Pattern\ Condition:\ \Delta s \approx \Delta f + \Delta b$ (3)

Formulas to estimate Δf_s and Δf_d

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

TABLE I
THREE DIFFERENT PATH PATTERNS

above; (ii) *fsd probing scenario* which uses *f*, *s*, and *d* probes; and (iii) *fsb probing scenario* which uses *f*, *s*, and *b* probes.

The condition of each path pattern can be represented by a simple equation that describes a long term relationship among Δf, Δb, Δs, and Δd, as shown in Table I. Based on the equation, we can efficiently check whether each path pattern is well matched and thereby choose a suitable path pattern to estimate Δf_s and Δf_d for each concerned intermediate node. We need not resolve the actual routes of paths that the four probes travel and the actual locations of repetitively congested links on these paths (and doing so could be very hard[1]). Instead, we simply check whether the queuing delays measured by the four probes satisfy the equation in the long term. In Table I, we summarize all three path patterns. For each of them, we show an exemplified scenario, the equation of pattern condition, and the formulas used to estimate Δf_s and Δf_d.

In practice, it is possible that none of the three path patterns can be matched. We therefore define a fourth path pattern for this case. We denote it *2-p probing scenario* in terms that we use only two types of probing packets (the *f* and *s* probes) to estimate Δf_s and Δf_d. The 2-p probing scenario is a pseudo path pattern. It is unconditionally matched. The corresponding formulas used to estimate Δf_s and Δf_d are:

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

Selecting probing techniques. For each path pattern, we call the method used to estimate Δf_s and Δf_d (including both the probing method and formulas) a *probing technique*. We refer to each probing technique by the name of the corresponding path pattern but omitting the word "scenario". We therefore have the following four probing techniques: *4-p probing*, *fsd probing*, *fsb probing*, and *2-p probing*.

For each intermediate node along the forward path, we select a suitable probing technique such that we can estimate Δf_s and Δf_d most accurately. Different probing techniques provide different accuracies. The 4-p and fsd probing can provide the tightest bounds for Δf_s and Δf_d, hence the highest accuracy; the fsb probing gives slightly looser bounds than the previous two; and the 2-p probing gives the loosest bounds though it can be unconditionally used.

To quantify the extent to which the path patterns for 4-p, fsd, and fsb probings are matched, we define *quality of measurability* (*QoM*) for each path pattern as shown below:

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

A QoM represents how well a path pattern is matched. It takes a value between 0 and 1. The larger it is, the better a path pattern is matched. By evaluating the average QoM over a longer time period, we can decide whether each probing technique is applicable. We then select a probing technique that can give the highest accuracy among the applicable ones. If none of the three techniques is applicable, we select the 2-p probing technique. The selection of probing techniques might need to be changed over time due to changes of underlying path conditions. By keeping track of QoMs, we adjust probing techniques online adaptively.

Pairing up the s and d probes. When using the 4-p or fsd probing technique, we must first pair up the *s* probe (to the concerned intermediate node) with a *d* probe (to an intermediate node on the backward path). However, pairing s and d probes is a non-trivial task. This is because an *s* probe can have several candidate *d* probes to pair up with, but different pairings could give different measurement accuracies. In addition, the optimal pairing can shift over time due to changes of underlying path conditions. We therefore develop an adaptive pairing algorithm to accomplish this task. This algorithm keeps track of performance history for pairings, including: (i) how many times a pairing has been selected; (ii) what was the average duration that a pairing "worked" during each time it was selected; (iii) what was the average QoM a pairing achieves. The algorithm optimizes pairings online by giving priority to pairings that were less frequently selected,

[1]For example, it is hard to resolve the route of an ICMP response from an intermediate node.

achieved longer durations, and had larger average QoMs.

B. Locating Congested Links

We can perform coordinated probing for each intermediate node (that responds to TTL-limited probes) on the forward path to estimate corresponding half path queuing delays Δf_s and Δf_d. By correlating half path queuing delay results of neighboring intermediate nodes, we can effectively locate congestion points at the granularity of a single link[2].

Deducing half-path congestion status. We first translate half path queuing delay estimates to corresponding congestion status. This is done by comparing queuing delay estimates with a proper threshold. (We will describe in detail how we get this threshold in Section III-A.) We represent congestion status using two measures — *congestion probability* and *confidence*. The congestion probability indicates the probability that the actual queuing delay is above the threshold. The confidence represents extent of certainty for the inferred status. Take the half path corresponding to Δf_s for example, we denote by P_{fs} its congestion probability. If the lower bound of Δf_s is above the threshold, we set P_{fs} to 1. Likewise, if the upper bound of Δf_s is below the threshold, we set P_{fs} to 0. In both cases, we set the confidence to 1 to indicate the 100% certainty. Otherwise, both P_{fs} and the confidence take values between 0 and 1 which are calculated as functions of the lower and upper bound of Δf_s, and the threshold.

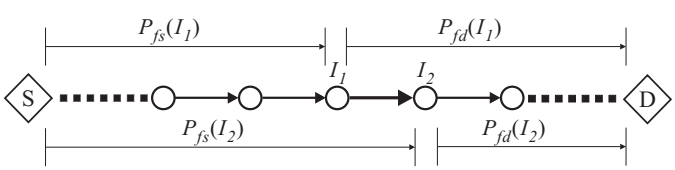


Fig. 2. Inferring congestion location

Inferring congestion location. When congestion probability is greater than 0.5, we regard a half path as congested, otherwise not congested. By correlating inferred half path congestion status of neighboring intermediate nodes, we can locate congested links on a path. Consider two consecutive intermediate nodes I_1 and I_2 on the forward path as shown in Figure 2, we regard the link between them as congested when the following conditions are satisfied[3]:

$$P_{fd}(I_1) > 0.5 \; and \; P_{fs}(I_2) > 0.5, \quad (4)$$
$$P_{fs}(I_1) \leq 0.5 \; or \; P_{fd}(I_2) \leq 0.5. \quad (5)$$

In practice, we send coordinated probes (to all intermediate nodes) at a rate of 2 Hz. At this rate, we could not capture all instantaneous congestion moments (each of them is in form of a continuous queue building-up and draining period). However, for a link exhibiting repetitive congestion, we can identify it using probing statistics of a relatively long period. To do this, we maintain a statistic measure *weighted congestion count* (*WCC*) for each link. Each time a link is detected to be congested by the above method, we increase the WCC by 1 if both nodes of the link show confidence = 1 for congestion status of all four half paths. If congestion status of some half paths has confidence $\neq$ 1, the increment will be a value between 0 and 1 computed based on confidences. Intuitively, the increment of WCC quantifies extent of certainty for the detected congestion. In addition, we double the increment if the following condition is satisfied:

$$P_{fs}(I_1) \leq 0.5 \; and \; P_{fd}(I_2) \leq 0.5. \quad (6)$$

Condition (6) is stronger than condition (5), and when satisfied it indicates a much higher extent of certainty for the detected congestion.

Identifying congested links — congestion positive rate. We define the measure *congestion positive rate* (*CPR*) to represent congestion status of each link in a simple way. The CPR indicates the congestion probability of a link at a moment. Its value is translated from WCC statistics. It uses WCC history of up to one minute. Under desirable path conditions, we can identify a congested link using the CPR almost immediately after congestion arises. Nevertheless, we still have a good chance to identify congestion under undesirable conditions by examining the relatively long measurement history. To provide a concrete idea, we show the algorithm that we use to set the CPR in our implementation below. Parameters used in this algorithm are refined via Internet-based experiments.

	if (WCC of recent 10 seconds $>$ 1.5)	then set CPR = 1
else	if (WCC of recent 20 seconds $>$ 2)	then set CPR = 0.9
else	if (WCC of recent 30 seconds $>$ 2.5)	then set CPR = 0.8
else	if (WCC of recent 40 seconds $>$ 2.75)	then set CPR = 0.65
else	if (WCC of recent 50 seconds $>$ 3)	then set CPR = 0.5
else	if (WCC of recent 60 seconds $>$ 3.25)	then set CPR = 0.35
else	set CPR = 0	

Tracing congested links — congestion intensity. Based on the CPR, we can further trace congestion status on each link. To do this, we send the end-to-end probe on the forward path, *i.e.*, the *f* probe, at a fast rate (10 Hz in our implementation). Each fast rate *f* probe can tell whether the path is congested or not. If the path is congested, we infer the location of congestion using the CPR. A link on the path that has a nonzero CPR at this moment is regarded as congested.

We generate statistics for traced results every 30 seconds and quantify the statistics by a measure — *congestion intensity*. This measure represents how frequently a link is congested during each 30-second-long time period. It is computed as the percentage of the fast rate *f* probes that report the link as congested within the 30 seconds.

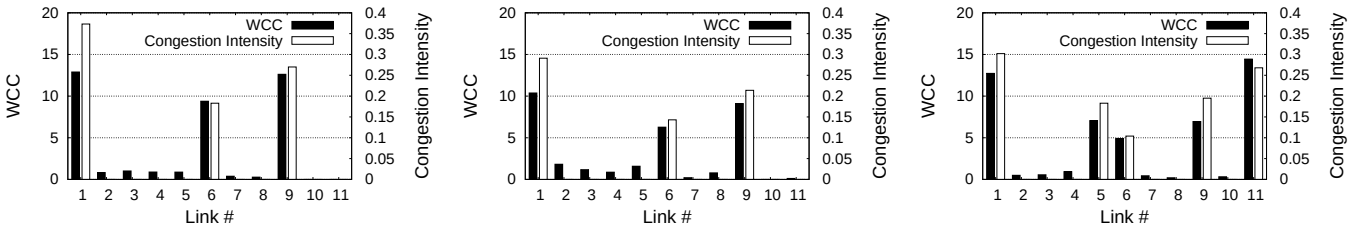
One subtlety in tracing congested links is that in some cases there could be multiple concurrently congested links when a fast rate *f* probe observe congestion on the path. For such an *f* probe, we assign a smaller weight when computing the congestion intensity of each link in order to reflect the uncertainty about the congestion location. In our implementation, we use the instantaneous CPR of each link as the weight for such moments.

C. A Measurement Example

Here we show measurement results of an experiment on the Emulab testbed [1] to exemplify the efficacy of our measure-

[2]Here "link" means IP level link. In addition, if some intermediate nodes do not respond to TTL-limited probes, the "link" is interpreted as a path segment between two closest intermediate nodes that respond to TTL-limited probes.

[3]Conditions for the first link and the last link are special cases and are much simpler. They are $P_{fs}(I_2) > 0.5$ and $P_{fd}(I_1) > 0.5$ respectively.



(a) 3 congested links at first (b) After adding 2 backward congested links (c) After adding 2 forward congested links

Fig. 3. Measurement results of an Emulab experiment in terms of *weighted congestion count* (WCC) and *congestion intensity*

ment methodology. We will demonstrate its high measurement accuracy even for cases of multiple concurrently congested links and its capability to decouple forward congestion from backward congestion.

In this example, we emulate a path consisting of 11 links. Each link has a capacity of 100 Mbps and a delay of 2 ms. We run our implemented measurement tool *Pong* (written in C++ and run on Linux) on the two endpoint machines. We use periodic on/off (50% time on, 50% time off) TCP cross traffic with different periods (the periods ranging from 0.5 to 2 seconds) to emulate concurrent but independent congestion on different links of the path.

First, we create congestion at links 1, 6, and 9 in the forward direction. Figure 3(a) plots the measurement results, which shows the *weighted congestion count* (WCC) and *congestion intensity* of each link for a 30-second-long time period. As we can see, although the WCC shows slight false positives on some non-congested links, the errors can be easily filtered. As a result, the congestion intensity shows no such false positives and it correctly identifies all three congested links.

Next, we add two additional congested links (links 2 and 5) in the backward direction. Figure 3(b) plots results for this case. As we can see, the false positives on non-congested forward links become larger than the previous case due to interferences from the two congested links in the backward direction. But the errors are still small and can be filtered. Therefore, the congestion intensity still accurately reflects congested links with no false positives. This case demonstrates our measurement methodology's capability to decouple forward congestion from backward congestion.

Finally, we remove the two backward congested links and add two more forward congested links (links 5 and 11). As we can see from the result shown in Figure 3(c), we successfully identify all the five congested links. This case demonstrates our methodology's high accuracy in locating congested links even when there are quite a number of concurrently congested links interfering with each other's congestion observation.

III. Optimizing Pong in the Internet

Measurement environment in the Internet is much more complex than that of a controlled testbed like the Emulab. For example, queuing delays reflecting the congestion on a link can differ substantially among different links, clock skews and jumps can happen at end hosts, route alterations may occur on paths, and ICMP responses could be queued at routers. All of these can significantly affect the performance of a tool. In this section, we addresses these practical measurement issues and we optimize our measurement tool's performance in reality.

A. Setting Queuing Delay Threshold

In our methodology, we compare the half path queuing delays measured by the coordinated probing with a threshold to decide whether each half path is congested or not. This threshold is set for each path dynamically during measurements; setting it correctly is critical for Pong's measurement accuracy. Still, this is a non-trivial task. In practice, links constituting an Internet path could be quite heterogeneous. For some links, queuing delays of several milliseconds could indicate severe congestion, while for other links such queuing delays are just trivial. In particular, when queue building-up events happen concurrently on two heterogeneous links of the same path, the one with a higher queuing delay scale can blur our observation of queue building-up on the one with a smaller queuing delay scale.

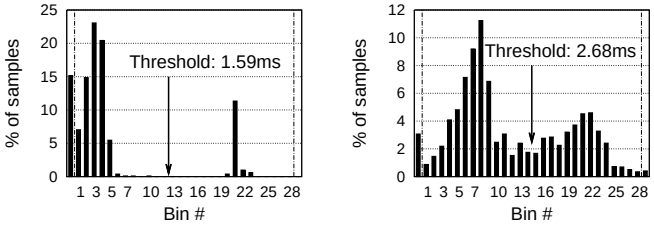
In this section, we introduce a best effort algorithm that we use to set the threshold in cases when the queue building-up on a link is not severely blurred.[4] This algorithm sets the threshold based on the statistical distribution of measured queuing delays on a path.

To represent the distribution, we use the following bin settings. We distribute queuing delay samples into 30 bins. The first bin corresponds to queuing delays < 0.15 ms and the last bin corresponds to queuing delays ≥ 300 ms. The middle 28 bins span the range from 0.15 ms to 300 ms with logarithmically uniform bin sizes. We set the bin sizes logarithmically uniform because the larger the average queuing delay, the larger the queuing delay deviation could be. The parameters 0.15 ms and 300 ms are refined through measurement experiments on a large number of Internet paths and also by accounting for the queuing-delay scales reported by others [2]–[4].

Since we focus on repetitively congested links, we expect the queuing delay distribution to show clustering effects. A typical case under desirable conditions is that we can observe two strong modes from the queuing delay distribution as shown in Figure 4. One mode corresponds to samples observing no congestion and the other corresponds to samples observing congestion. In such a case, we can easily set a proper threshold in between the two modes, as shown in the figure.

However, in some cases it could be much harder to identify the clustering for congested queuing delay samples than the

[4]For scenarios when the observation is severely blurred, we introduce a separate algorithm in Section III-C.



(a) The Emulab path in Section II-C (b) An Internet path in practice

Fig. 4. Queuing-delay distribution with two strong modes

above case. Regardless, our algorithm makes its best effort to infer this clustering and set a threshold right below the congested cluster. It does so by sequentially checking the queuing delay distribution with four major patterns that we have refined via a large number of Internet experiments. If a pattern is matched, it uses a method corresponding to the pattern to set the threshold. Table II highlights this algorithm.

Distribution pattern	Method used to set the threshold
Pattern A: More than 90% samples fall in the first bin (bin 0).	Locate a proper threshold from bin bin 1 to bin 4.
Pattern B: Samples exhibit two relatively strong modes from bin 1 to bin 28.	Locate an appropriate valley point between the two strong modes as the threshold.
Pattern C: More than 15% samples fall in bin 0, and less than 3% samples fall in the last bin (bin 29).	Search from bin 1 towards bin 28 for a foot point (starting from which the per-bin number of samples descends to a considerably low value).
Pattern D: None of the above patterns is matched.	First locate a peak bin from bin 1 to bin 28. Then search from the peak bin towards bin 28 for a foot point.

TABLE II
ALGORITHM TO SET QUEUING DELAY THRESHOLD

B. Minimizing Measurement Errors

Our methodology makes best effort to minimize measurement errors. We use the Linux kernel level timestamps (instead of the application level timestamps) when implementing the coordinated probing to improve its measurement accuracy. However, there are still many interference factors in practice that can cause measurement errors. Such factors include router alterations, clock skews and jumps at end hosts, ICMP queuing at routers, and other anomalies. We carefully handle these factors in our methodology by detecting and reacting to them.

We can explicitly detect router alterations, clock skews, and clock jumps. Router alterations are detected via traceroute-like methods. Clock skews and jumps are detected by keeping track of measured minimum one-way delays for the forward and the backward paths. (For example, a continuous decreasing of a minimum one-way delay indicates a clock skew.) When significant router alterations, clock skews, or clock jumps are detected, we pause the measurement and keep track of the anomaly until it is relieved. If it is just a transient anomaly, we simply filter out the affected measurement samples.

We can implicitly detect ICMP queuing and many other unidentified anomalies that affect measured queuing delays by tracing the *quality of measurability* (QoM) measure used in the coordinated probing. The QoM indicates the matching extent of a path pattern for measured queuing delays. When such anomalies happen, the QoM usually shows an exceptionally small instantaneous value. We can then use this instantaneous QoM value as a coefficient when updating measurement results from the sample. In this way, samples affected by such anomalies will have very limited impact on the final results.

C. Quantifying Measurement Accuracy

Although our methodology makes the best effort to measure the congestion location on a path, its measurement accuracy could differ a lot for different paths due to underlying path conditions. However, a link, especially a link in the Internet core, can be observed by many paths which can give very different measurement accuracies for congestion on the link. We therefore can select a path that provides the best accuracy when measuring congestion on a specific link.

To this end, we quantify measurement accuracy with a measure — *link measurability score*. The link measurability score represents the measurement accuracy for congestion on a specific link when measured from a specific path. The higher the value, the better the quality. It is computed based on the following three components:

- Node score, which takes into account the path pattern matched for the coordinated probing of an intermediate node (since different path patterns give different measurement accuracies for half path queuing delays) and the corresponding QoM achieved (which quantifies the matching extent of the path pattern and also captures extent of transient anomalies). For each link, node scores of both nodes of the link are considered. This measure affects the link measurability score the most.
- Queuing threshold score, which takes into account the quality of the queuing delay threshold set for the path. As described in Section III-A, we use a best effort algorithm to set the queuing delay threshold by matching the queuing delay distribution with four patterns. However, different patterns result in different qualities for the threshold, thereby leading to a different queuing threshold score.
- Observability score. Our measurement experience shows that congestion observed on a less frequently congested link can be blurred by a much more frequently congested link on the same path. We therefore assign a small observability score for those less frequently congested links to indicate the uncertainty for their measurement results when we detect a much more frequently congested link in *recent* time on the same path.

D. Evaluation-based Tuning

We evaluate our methodology using the Emulab testbed where we can control the ground truth of congested links. However, to address the practical issues discussed above, we resort to the Internet-based evaluation. We use the PlanetLab testbed [5] for our Internet-based experiments. We deploy our implemented measurement tool *Pong* on over 300 PlanetLab hosts from which we have performed measurement experiments on tens of thousands of Internet paths. These experiments help us empirically tune parameters used in our methodology: (i) the QoM threshold above which we regard a path pattern as matched, (ii) the parameters used to

translate the queuing delay of a half path to the corresponding congestion probability and confidence, (iii) the parameters used to compute the link measurability score, *etc*.

A big challenge for the Internet-based evaluation is the difficulty to get the ground truth of congested links. But given that we are measuring repetitively congested links, we stand a good chance to infer the ground truth by exploiting the temporal locality of congestion. In addition, since the same congested link can be observed from different paths, we exploit measurement consistency among these paths to help infer the ground truth, as we explain in detail below.

IV. Performance Analysis

In this section, we analyze Pong's performance in the Internet by evaluating the following aspects: (i) Pong's self-consistency on measured congestion locations among independent measurements, (ii) the performance of coordinated probing in terms of probing technique utilization, (iii) Pong's overall measurement accuracy in terms of the link measurability score, and (iv) advantages of the link-level congestion monitoring capability enabled by Pong.

A. Self-consistency Validation

We evaluate Pong's measurement accuracy using self-consistency validation. In essence, we want to confirm that we can get consistent congestion observations when measuring the same repetitive congested link from multiple vantage points. 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 methodology.

Here, we present the self-consistency validation results in one of our Internet experiments. This is a PlanetLab based experiment, in which we have measured over 20,000 different paths within 10 days using over 300 PlanetLab hosts. Each path is measured for 100 minutes. From the raw measurement data, we collect a total of 346,591 congestion events. Each congestion event is a continuous congestion period on a link during which measured congestion intensity is consistently above a threshold. Based on these congestion event samples, we collect data useful for self-consistency validation in the following way.

First, we select the congestion samples that reside exclusively on *individual* IP-level links for which both ends responds to TTL-limited probes. In this way, we guarantee that we know the accurate location (*i.e.*, on the granularity of a single IP-level link) for these congestion samples. This leaves us with 335,391 remaining samples.

We then refine the data set by filtering 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. Thus, to make our validation more convincing, we prefer to use self-consistency results on links for which the congestion scale is relatively small. Second, links in the core are more likely to be measured by endpoints from different physical locations. Therefore, data on these links are more suitable for self-consistency validation.

After filtering 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 that 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 select the eight links that are shared by the most paths. Table III provides 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	# of Paths	# of Sources	# of Destinations
Link 1	312	3	140
Link 2	301	57	35
Link 3	295	19	69
Link 4	294	32	38
Link 5	262	120	3
Link 6	252	54	31
Link 7	243	6	110
Link 8	239	14	101

TABLE III
Top Eight Congested Links Observed by Most Paths

For each of the eight links, we check related measuring paths by examining sets of link congestion status graphs generated by Pong. We find that Pong locates congested links (that exhibit repetitive congestion) with very high accuracy. There are almost zero false positives on the links close to the shared link. For the vast majority of these measuring 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 the temporal consistency for the congestion measured on shared links. To accomplish this, we collect concurrent measurements, *i.e.*, overlapping measurement time periods for paths associated with a shared link. Then, we plot and manually examine these periods. We find that, for results annotated with relatively good link measurability scores, the measured congestion shows high temporal consistency: during an overlapping measurement period, if on one path we observe congestion for the shared link, we can also observe it on other paths — at the same time and with similar congestion scales and time patterns.

B. Probing Technique Utilization

As we described in Section II-A2, when conducting coordinated probing, Pong selects probing techniques online based on matched path patterns. The probing techniques actually in use are crucial for the measurement accuracy. The *4-p*, *fsd*, and *fsb* probing techniques can provide high measurement accuracies, while the measurement accuracy of the *2-p* probing technique is relatively low. In this section, we present our measurement statistics on the utilization of each probing

technique in Internet experiments. This helps us get a better understanding on Pong's actual performance in practice.

The statistics shown here corresponds to the same PlanetLab based experiment described in the previous section, which has measured over 20,000 different paths within 10 days. We log selected probing techniques at each intermediate node on each path every 30 seconds. Based on these logged probing technique samples, we get the following utilization statistics.

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 the fsd probing is only utilized 1% of time approximately. The large percent for the 4-p and 2-p probing is not a surprise. As explained in Section II-A2, 4-p probing has the highest priority and therefore is always tried first, and 2-p probing is the last resort. The fsd and fsb probing are applied only as transient techniques between the default ones. Therefore, their percent is not high. Particularly, for a large majority of cases where the fsd probing is applicable, we can also find an appropriate pairing to make the 4-p probing work. Therefore, the 4-p probing is finally used. This is exactly why the percent of the fsd probing is the lowest.

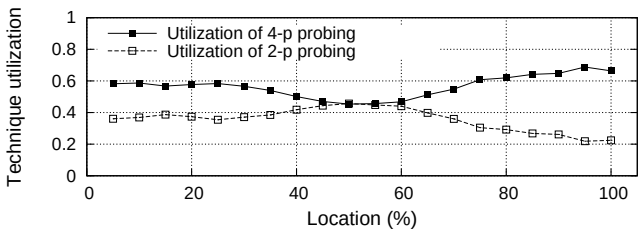


Fig. 5. Probing technique utilization vs. location

Figure 5 shows the utilization of the 4-p and 2-p probing techniques relative to the location of intermediate nodes. The location is represented using the normalized hop number of an intermediate node in the format of a percent number, *i.e.*, 0% means the first intermediate node on the path, closest to the source, and 100% means the last intermediate node on the path, closest to the destination.

The figure shows that the 4-p and 2-p probing techniques prevail: their sum is typically above 90%. However, the spatial distribution is not uniform. For the intermediate nodes closer to edges the 4-p probing prevails, which means there is a high opportunity to find a suitable *d* probe to an intermediate node on the backward path to pair up with the *s* probe to the concerned intermediate node on the forward path. For the concerned intermediate nodes closer to the core (*i.e.*, the 50% location), the opportunity of such pairing decreases. This is because forward and backward paths are typically disjoint in the core [6]. However, the figure also shows that many (over 45%) of the intermediate nodes on *disjoint* paths in the core could still be successfully paired.

C. Measurement Accuracy for Non-access Links

In this section, we analyze Pong's measurement accuracy for the target that its methodology is optimized for — non-access links that exhibit repetitive congestion. We present the statistical distribution of measurement accuracies achieved on such links based on our Internet experiments. The measurement accuracies are indicated by the link measurability score (LMS) that we introduced in Section III-C.

The statistics shown here corresponds to the same PlanetLab based experiment described in the previous section. In this experiment, we have collected about 24,000 congestion samples on non-access links that exhibit repetitive congestion. Each sample corresponds to 30-second-long time slot and is annotated with a congestion intensity and an LMS. Recall that the congestion intensity represents how frequently we observe congestion on a specific link during a 30-second-long time slot. Thus, we can select samples of repetitive congestion by filtering out samples showing small congestion intensity. In addition, we can select samples for non-access links based on a link's normalized location on a path. We roughly treat a link with a normalized location between 20% and 80% as a non-access link. In this way, we collected the above 24,000 samples.

In our implementation, the LMS takes a value between 0 and 6. Before explaining the LMS distribution, we first introduce the implications of major LMS levels. These implications are learned through our evaluation-based tuning described in Section III-D.

"LMS=0" means both nodes of the link are using 2-p probing technique. We have observed end-to-end congestion on the path at a moment, but Pong is unable to accurately locate this congestion. The conclusion that the link is congested may not be reliable. "LMS=1" usually means at least one node of the link is using 4-p or fsd probing technique, but the quality of measurability (QoM) is not high. This is the lowest level that we consider the measurement accuracy as acceptable. "LMS=2" usually means that either (i) one node of the links is using 4-p or fsd and it achieves a high QoM or (ii) both nodes are using 4-p or fsd and have moderate QoMs. It indicates moderate measurement accuracy. "LMS=3" usually means both nodes are using 4-p or fsd and both achieve good QoMs. It indicates good measurement accuracy. "LMS$\geq$4" indicates very good measurement accuracy.

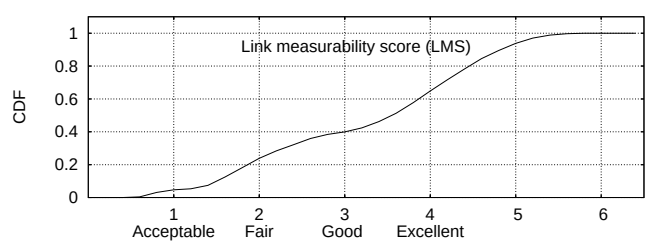


Fig. 6. CDF of link measurability score

Within the aforementioned 24,000 congestion samples, we find 63% of them associate with non-zero LMS. Figure 6 shows the cumulative distribution function (CDF) of LMS for these samples with non-zero LMS. As we can see, (i) 95% of these samples have an acceptable measurement accuracy (LMS$\geq$1), (ii) 75% have a better-than-fair measurement accuracy (LMS$\geq$2), (iii) 60% have a good measurement accuracy (LMS$\geq$3), and (iv) 35% have an excellent measurement accuracy (LMS$\geq$4).

The LMS distribution helps us understand the Pong's measurement accuracy in reality. Meanwhile, it also reflects that there is a good chance for Pong to achieve a good measurement accuracy in practice even when we are restricted to use a designated path. Given that the same non-access link can usually be measured from many different paths, we stand a good chance to achieve a high measurement accuracy by selecting a suitable measuring path.

D. Advantages of Link-level Congestion Monitoring

Due to its very lightweight nature, Pong could be effectively used for long term congestion monitoring in addition to on demand congestion measurement. For example, for a path consisting of 10 intermediate nodes, its *maximum* probing overhead from either endpoint is only 2.2 kBps. Moreover, when compared to pure end-to-end monitoring approaches, the link-level congestion monitoring enabled by Pong has apparent advantages:

First, link-level congestion monitoring allows us to focus on congestion at a specific location and pinpoint congestion at the level of a single link. On the contrary, the pure end-to-end congestion monitoring will inevitably mix congestion from all locations on a path. Secondly, when monitoring a wide network area, link-level congestion monitoring makes it possible to cover the area in an unbiased way. On the contrary, a location-unaware end-to-end monitoring approach could inevitably cover some congested links with many more measuring paths than some other links. Such biased coverage will lead to biased measurement results because congestion on the former links is repeatedly counted for many more times than that on the latter links.

V. Related Work

Active probing techniques used in measurement tools can be divided into two categories: end-to-end approaches (*e.g.*, [7]–[12]) and router-response-based approaches (*e.g.*, [13]–[15]). End-to-end approaches send single probe packets, packet pairs, or packet trains along the whole path. They measure one-way delay or packet dispersion and infer end-to-end path properties such as bottleneck capacity [8], [12], [16], available bandwidth [7], [11], [17], and bottleneck location [10], [17], [18]. 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 measuring link-level queuing delays to infer congestion locations. Pong sends both end-to-end and router-response-based probes and correlates them in a novel way. Unlike end-to-end tools such as *Pathneck* [10], *Stab* [17], and *BFind* [18] which can only locate the dominant bottleneck or minor bottlenecks in front of the dominant one, Pong can measure multiple concurrently congested points on the same path. Unlike router-response-based tools such as *Tulip* [15] and *Pathchar* [14], 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 [18] exploits TCP's property of gradually filling up the available bandwidth and combines it with router-response-based probes to measure the location of dominant bottlenecks. 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. On the contrary, the Pong's overhead is very low, which makes it suitable for continuous monitoring in the long term. *Pathneck* [10] and *Stab* [17] are two other examples that integrate available bandwidth measurement approach with a router-response-based probing method to locate dominant bottlenecks. Although the overhead for each of them is significantly lower than that of *BFind*, it is still much higher than that of Pong because they rely on relatively long packet trains when probing the available bandwidth.

Gurewitz and Sidi have performed a mathematical study on estimating one-way delays from round-trip delay measurements [19]. 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. On the contrary, since Pong measures queuing delays instead of the absolute delays, it can easily mitigate the two problems — when inferring underlying path conditions, Pong does not rely on knowledge of the actual topological characteristics of the paths that probes travel; also, the desirable Pong conditions have a high probability of being satisfied in practice, as we demonstrated in the previous section. In addition, the approach of [19] requires measurements from a large number of vantage points, while Pong only relies on measurements from the two endpoints of a path.

Network tomography approaches [20]–[25] exploit temporal correlations among results observed by 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* [26], has developed an unbiased algebraic model that significantly reduces the complexity of network tomography and improves its inference accuracy. However, even when equipped with *Mils*, a network tomography approach is still more complex relative to Pong. Pong shares a flavor of network tomography approaches in that it also exploits temporal correlations of congestion measured by probes traveling shared path segments. Still, Pong is much simpler and much easier to deploy than a network tomography tool, especially when we want to generate measurement results in real time.

Network radar [27], [28] 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 the radar's vulnerability to interferences from congestion on shared path segments on returning 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. Although Pong requires collaboration from the receiver side, this restriction is not dramatic when our concerned targets are congested links in the Internet core, which can be well measured from a relatively small number of vantage points that we can control.

Detecting shared congestion of multiple paths is another related work. For example, Rubenstein *et al.* [29] detect shared points of congestion by exploiting Poisson probes; Kim *et al.* [30], [31] detect shared congestion by using a wavelet denoising approach. Because both approaches strictly leverage end-to-end probing, they have to rely on advanced probing patterns and signal processing methods to identify shared congestion. In contrast, by combining end-to-end probing with router-response-based probing, Pong provides an efficient way to directly locate congested points which can potentially lead to a much simpler method to detect shared congestion.

VI. Conclusions

In this paper, we presented *Pong*, a measurement tool capable of accurately locating and monitoring congested links that exhibit repetitive congestion. By exploiting coordinated probing from both endpoints of a path, Pong can detect and monitor congestion on the Internet core links which are typically much harder to measure than edge links. Pong can locate congested links almost immediately for significant congestion (in terms of both queuing delay scales and the period of continuous queue building-up and draining epochs). Meanwhile, it can also accurately locate congested links that show small instantaneous queuing scales but exhibit repetitive congestion over longer time scales. The very lightweight nature of Pong makes it suitable for long term monitoring tasks as well.

Pong can (i) strategically combine end-to-end and router-response-based probes sent from both endpoints to improve congestion observability; (ii) effectively self-adapt to underlying topological characteristics to accurately locate congested links; (iii) detect and filter out periods of its own measurement inaccuracy; and (iv) quantify its measurement accuracy for a given link observable from multiple vantage points, allowing us to select suitable paths to optimize measurement performance.

Our experimental results on the Emulab testbed show that Pong can accurately locate congested links even in cases of quite a number of concurrently congested links on the same path. Moreover, it can successfully decouple congestion on the forward path from that on the backward path. Finally, our PlanetLab experiments expose Pong's high potential to achieve a high measurement accuracy in the real Internet environment.

Acknowledgements

This research is supported by NSF CAREER Award no. 0746360 and by Cisco Systems.

References

[1] "Emulab," http://www.emulab.net/.
[2] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales?" in *ACM SIGMETRICS*, Banff, Alberta, Canada, Jun. 2005.
[3] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "A measurement study of internet bottlenecks," in *IEEE Infocom*, Miami, FL, Mar. 2005.
[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] "Planetlab," http://www.planet-lab.org.
[6] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM ToN*, vol. 5, no. 5, pp. 601–615, Oct. 1997.
[7] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan, "PCP: Efficient endpoint congestion control," in *NSDI*, San Jose, CA, May 2006.
[8] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *IEEE INFOCOM*, Anchorage, AK, Apr. 2001.
[9] ——, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM ToN*, vol. 12, no. 6, pp. 963–977, 2004.
[10] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang, "Locating Internet bottlenecks: Algorithms, measurements, and implications," in *ACM SIGCOMM*, Portland, Oregon, Sep. 2004.
[11] 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.
[12] S. Saroiu, P. Gummadi, and S. Gribble, "Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments," in *IEEE INFOCOM*, New York, NY, Jun. 2002.
[13] K. Anagnostakis, M. Greenwald, and R. Ryger, "cing: Measuring network-internal delays using only existing infrastructure," in *IEEE INFOCOM*, San Francisco, CA, Jun. 2003.
[14] A. Downey, "Using Pathchar to estimate Internet link characteristics," in *ACM SIGCOMM*, New York, NY, Oct. 1999.
[15] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level Internet path diagnostics," in *ACM SOSP*, Bolton Landing, NY, Oct. 2003.
[16] 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.
[17] V. Ribeiro, R. Riedi, and R. Baraniuk, "Locating available bandwidth bottlenecks," *IEEE Internet Computing*, vol. 8, no. 5, Sep. 2004.
[18] A. Akella, S. Seshan, and A. Shaikh, "An Empirical Evaluation of Wide-Area Internet Bottlenecks," in *ACM SIGMETRICS*, San Diego, CA, Jun. 2003.
[19] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *IEEE INFOCOM*, Anchorage, AK, Apr. 2001.
[20] A. Adams et al, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications*, May 2000.
[21] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, New York, NY, 2002.
[22] N. Duffield, "Simple network performance tomography," in *ACM IMC*, New York, NY, 2003.
[23] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of Internet link lossiness," in *IEEE INFOCOM*, San Francisco, CA, Apr. 2003.
[24] 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.
[25] Y. Tsang, M. Coates, and R. Nowak, "Network delay tomography," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2125–2136, Aug. 2003.
[26] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, Pisa, Italy, Sep. 2006.
[27] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "Network radar: tomography from round trip time measurements," in *ACM IMC*, New York, NY, 2004.
[28] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak., "On the performance of round trip time network tomography," in *IEEE ICC*, Jun. 2006.
[29] 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.
[30] M. Kim, T. Kim, Y. Shin, S. Lam, and E. Powers, "A wavelet-based approach to detect shared congestion," in *ACM SIGCOMM*, Portland, Oregon, Sep. 2004.
[31] ——, "Scalable clustering of Internet paths by shared congestion," in *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.