PBProbe: A CapProbe based Scalable Capacity Estimation Tool*

Ling-Jyh Chen, Li Lao, Tony Sun, Guang Yang, M. Y. Sanadidi, Mario Gerla
Department of Computer Science, University of California at Los Angeles

Abstract

We present a scalable capacity estimation technique, called PBProbe. PBProbe is based on CapProbe. Instead of solely relying on packet pairs, PBProbe employs a "packet bulk" technique and adapts the bulk length in order to overcome a well known problem with packet pair based approaches, namely the lack of accurate timer resolution. As a result, PBProbe not only preserves the simplicity and speed of CapProbe, but it also correctly estimates link capacities within a much larger range. Using analysis, we evaluate the accuracy and speed of PBProbe with various bulk lengths. We then perform a set of experiments to evaluate the accuracy of PBProbe in the Internet over wired and wireless links. Finally, we perform emulation and Internet experiments to verify the accuracy and speed of PBProbe on extremely high-speed links. The results show that PBProbe is consistently fast and accurate in the great majority of test cases.

1 Introduction

Estimating the bottleneck capacity of an Internet path is a fundamental research problem in computer networking; knowledge of such capacity is critical for efficient network design, management and usage. In the past few years, with the growing popularity of emerging technologies such as overlay, peer-to-peer (P2P), sensor, grid and mobile networks, it is becoming increasingly desirable to have a simple, fast and accurate tool for capacity estimation and monitoring. Moreover, the tool must be scalable and applicable to a variety of network configurations.

A number of techniques have been proposed for capacity estimation on generic Internet paths [3, 7, 13, 14, 16, 18, 25]. Among them, CapProbe [16] and Pathrate [13] have been well accepted as two fast and accurate tools in generic network scenarios. However, CapProbe is a round-trip estimation scheme that works well only on paths consisting of a symmetric bottleneck link. Pathrate, on the other hand, is based on histograms and may converge slowly when the initial dispersion measurements are not of unimodal. As a result, CapProbe has difficulty estimating the capacities of asymmetric links [12], and Pathrate performs poorly on wireless links [16].

To address the problems above, specialized capacity estimation tools have been proposed for specific network scenarios. For instance, ALBP [22] and AsymProbe [12] are intended for capacity estimation on asymmetric links, AdHoc Probe [11] aims to estimate the end-to-end path capacity in wireless networks, etc. Yet, a general and practical capacity estimation tool applicable to any network configurations is still lacking. This technique must be simple, fast, accurate, minimally intrusive, and work well in all aforementioned scenarios including asymmetric, wireless and high speed links.

In this paper, we propose such a scalable capacity estimation tool called PBProbe. PBProbe is inspired by CapProbe. However, instead of relying on one pair of packets, PBProbe employs the concept of "Packet Bulk" to adapt the number of probing packets in each sample in accordance to the dispersion measurement. More specifically, when the bottleneck link capacity is expected to be low, PBProbe uses one pair of packets as usual (i.e. the bulk length is 1). For paths with high bottleneck capacities, PBProbe increases the bulk length and sends several packets together, in order to enlarge the dispersion between first and last packet and therefore overcome the timer resolution problem. As we will discuss later, this is the main challenge in the estimation of high capacity links [15, 17].

We study the performance of PBProbe under different loads of Poisson cross traffic via analytical models. We also evaluate the accuracy and speed of PBProbe, through testbed experiments, in various network configurations including generic Internet links, asymmetric links, and wireless links. We have carried out exten-

*This material is based upon work supported by the National Science Foundation under Grant No. CNS-0435515.

sive testbed experiments to validate the performance of PBProbe on high speed links; we have also compared it to Pathrate. The results show that PBProbe accurately and rapidly estimates the link capacities in all tested scenarios, thereby outperforming Pathrate in most experiments.

The rest of the paper is organized as follows. In section 2, we present an overview of CapProbe and summarize related work on capacity estimation. In section 3, we present and describe PBProbe. In section 4, we present an analysis of PBProbe and evaluate the speed and accuracy of PBProbe with Poisson cross traffic. In section 5, we present PBProbe experiments in general network scenarios. In section 6, we evaluate PBProbe on high speed links in our emulator testbed as well as on the Internet . Section 7 concludes the paper.

2 Background and Overview

2.1 Related Work

Previous research on capacity estimation relied either on delay variations among probe packets as illustrated in Pathchar [14], or on dispersion among probe packets as described in Nettimer [18] and Pathrate [13]. Pathchar-like tools (such as pchar [7] and clink [3]) have limitations in accuracy and speed as shown in [16] [19]. Moreover, they evaluate the capacity of a link based on the estimates on previous links along the path, thus estimation errors accumulate and amplify with each measured link [25].

Dispersion-based techniques suffer of other problems. In particular, Dovrolis' analysis in [13] showed that the dispersion distribution can be multi-modal due to cross traffic, and that the strongest mode of such distribution may correspond to either (1) the capacity of the path, or (2) a "compressed" dispersion, resulting in capacity over-estimation, or (3) to the Average Dispersion Rate (ADR), which is always lower than capacity. Another dispersion-based tool, SProbe [25], exploits SYN and RST packets of the TCP protocol to estimate the downstream link capacity, and employs two heuristics to filter out those samples which have experienced cross traffic. However, SProbe could not work properly when the network is highly utilized. [21].

Unlike the above approaches, CapProbe [16], uses both dispersion measurements and end-to-end delay measurements to filter out the packet pair samples that where distorted by cross traffic. This method has been shown to be both fast and accurate in a variety of scenarios. The original implementation of CapProbe uses ICMP packets as probing packets, and it measures the bottleneck capacity on a round-trip basis. As a result, the capacity estimate may fail when the path is asymmetric. Other difficulties are encountered when intermediate nodes employ priority schemes to delay ICMP packet forwarding (e.g. Solaris operating system limits the rate of ICMP responses, and it is thus likely to perturb CapProbe measurements) [26].

Recent capacity estimation studies have extended the target network scenarios to more diverse environments. For instance, Lakshminarayanan et al. have evaluated estimation tools of capacity and available bandwidth in the emerging broadband access networks [20]. Chen et al. extended CapProbe to estimate *effective path capacity* in ad hoc wireless networks [11]. In addition, ABLP [22] and AsymProbe [12] have been proposed for capacity estimation on the increasingly popular asymmetric links (e.g. DSL and satellite links).

Capacity estimation on high speed links is still a challenge. Though recent studies have verified the accuracy of Pathrate estimates on gigabits links [24], they did not include the experimental evaluation of such estimates. Moreover, the evaluation was done on an emulator-based testbed, which cannot represent realistic Internet dynamics.

In this paper, we propose a novel packet bulk technique for capacity estimation on high speed links and present a measurement tool called PBProbe. PBProbe is based on CapProbe, and it probes bottleneck link capacity using UDP packets (instead of ICMP packets as CapProbe does). We recap the CapProbe algorithm in the next subsection.

2.2 CapProbe Overview

CapProbe is a packet-pair based capacity estimation technique, which has been shown to be both fast and accurate over a large range of scenarios [16]. Conceptually speaking, when a back-to-back packet pair is launched into a network, and assuming they arrive at the bottleneck link unperturbed (i.e. back to back) by cross traffic on previous links, they are always dispersed at the bottleneck link according to the bottleneck capacity. If such dispersion arrives at a destination unperturbed, it will accurately reflect the bottleneck capacity (as shown in Fig. 1-c). However, if either packet in a pair has been queued due to cross traffic, the dispersion of this sample might be either expanded or compressed, where "expansion" of dispersion leads to under-estimation and "compression" of dispersion leads to over-estimation of the capacity (as shown in Fig. 1-a,b).

CapProbe combines the use of dispersion measurements and end-to-end delay measurements to filter out packet pair samples distorted by cross traffic. The basic idea is that, if a packet pair sample has not been queued by cross traffic, it will estimate the bottleneck capacity correctly. For such "good" samples, the sum of the de-

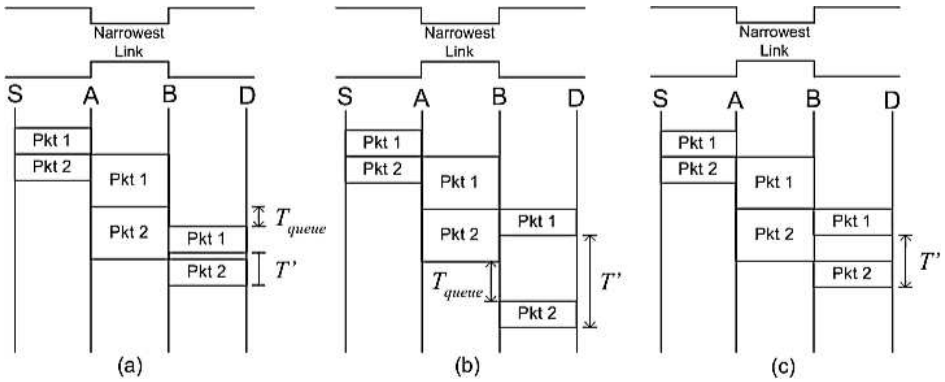

Figure 1: (a) Under-estimation caused by "expansion" (b) Over-estimation caused by "compression" (c) The ideal case.

lays of the packet pair packets, called the *delay sum*, does not include cross-traffic induced queuing delay, and is indeed smaller than the delay sum of the samples distorted by cross traffic. Thus, this kind of "good" samples can easily be identified since their delay sum will be the minimum among the delay sums of all packet pair samples, and we refer to their delay sum as the *minimum delay sum*. In this way, the capacity can be estimated by the equation:

$$C = \frac{P}{T} \tag{1}$$

where P is the sampling packet size, and T is the dispersion of the sample packet pair with the minimum delay sum.

However, since CapProbe relies on dispersion measurements for accurate capacity estimation, it suffers inaccuracy in estimating high speed links and/or when operating on slow machines [17]. More specifically, according to Eq. 1, when C is extremely large, either T must become small or P must become very large. Since the accuracy of T measurement is limited by the system timer resolution, the only feasible solution for CapProbe estimation on high speed links is to enlarge the packet size. However, once the packet size becomes larger than the Maximum Transmit Unit (MTU), this "big" packet will be segmented into several fragments before entering the network, and these fragments will be reassembled to the original packet size at the receiver [23]. The latency caused by segmentation (i.e. on the sender) and reassembly (i.e. on the receiver) leads to expansion of the dispersion measurement and therefore results in under-estimation. In order to prevent such additional latency, we propose to send multiple packets back to back, each with MTU packet size, together (i.e. virtually, the probing packets are launched into the network as a "big" packet). As a result, the dispersion measurement is applicable to accurate capacity estimation using CapProbe algorithm since the segmentation and reassembly latency can be avoided. We present the design, analysis, and evaluation of PBProbe in the following sections.

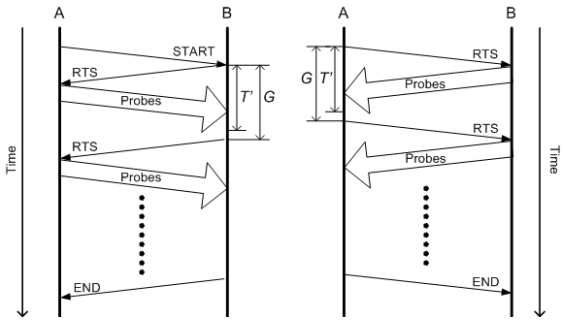

Figure 2: Illustration of PBProbe (a) Phase I: measuring forward direction link capacity; (b) Phase II: measuring backward direction link capacity.

3 Proposed Approach: PBProbe

In this section we introduce PBProbe. Similar to CapProbe, PBProbe estimates the link capacity by actively sending a number of probes to the network and using the *minimum delay sum* filter to identify the "good" sample. However, instead of employing a packet pair, PBProbe uses *packet bulk* of length k in each probing and measures the capacity for each direction separately. Specifically, there are two phases in PBProbe. In the first phase, PBProbe estimates the capacity of the *forward* link; whereas in the second phase, PBProbe estimates the capacity of the *backward* link. Fig. 2 illustrates the algorithm of PBProbe.

In the first phase (as shown in Fig. 2-a), host A first

sends one *START* packet to host B to initiate the estimation process. Once the process is initiated, B sends a *Request To Send (RTS)* packet to A every G time units. Upon the receipt of the *RTS* packet, host A immediately sends B a packet bulk of length k (note: bulk length = k means that $k+1$ packets are sent back to back). For the i-th probing sample, suppose B sends the *RTS* packet at time $t_{send}(i)$ and receives the j-th packet (in the i-th sample) at time $t_{rcv}(i,j)$. The delay sum (i.e. S_i) and the dispersion (i.e. D_i) of the i-th packet bulk sample are given by:

$$S_i = (t_{rcv}(i,1) - t_{send}(i)) + (t_{rcv}(i,k+1) - t_{send}(i)) \quad (2)$$

$$D_i = t_{rcv}(i,k+1) - t_{rcv}(i,1) \quad (3)$$

If neither the first nor the $(k+1)$th packet has experienced cross-traffic induced queueing, the sample will reflect the correct capacity. Thus, the "good" sample (say, the m-th sample) is identified by applying the minimum delay sum filter to all probing samples (say, n samples):

$$m = arg \min_{i=1...n} S_i \quad (4)$$

Therefore, the capacity estimate is made by using the dispersion of the m-th sample with the minimum delay sum:

$$C = \frac{kP}{D_m} \quad (5)$$

where P denotes the packet size of each probing packet. Since the packet bulk samples are delivered only in the forward direction, the estimated capacity corresponds to the bottleneck in this direction.

Once the first phase ends, B sends an *END* packet to A, and PBProbe enters the second phase (as shown in Fig. 2-b). In this phase, A first sends an *RTS* packet (every G time units) to B, and B replies a packet bulk of length k right upon the receipt of each *RTS* packet. Similar to the first phase, PBProbe measures the delay sum and dispersion for each sample, and estimates the capacity in the backward direction by using the minimum delay sum filter.

3.1 The Inter-sample Period: *G*

PBProbe probes the link capacity by sending a packet bulk every G time units. The value of G is critical for the convergence time of PBProbe. The larger G is, the slower PBProbe estimation becomes. However, G can not be too small either. If it is too small, PBProbe is more likely to create congestion in the network. Therefore, in PBProbe, we set G to be:

$$G = \frac{2D_{m'}}{U} \quad (6)$$

where $D_{m'}$ is the dispersion of the good sample (i.e. $D_{m'} = \frac{kP}{C}$), which has the minimum delay sum among all probing samples seen so far, and U is the maximum network utilization allowed for PBProbe estimation.

Corollary 1. *If $G = \frac{2D_{m'}}{U}$, PBProbe will never utilize the network with utilization larger than U.*

Proof. Since $k \geq 1$, we know that

$$G = \frac{2D_{m'}}{U} \geq \frac{D_{m'}}{U} + \frac{D_{m'}}{kU} = \frac{kP}{CU} + \frac{P}{CU} = \frac{(k+1)P}{CU} \quad (7)$$

$$\Rightarrow \frac{(k+1)P}{G} \leq CU \quad (8)$$

Let R denotes the introduced data rate of the packet bulk probing, i.e. $R = \frac{(k+1)P}{G}$; therefore we can conclude $R \leq CU$, i.e. the probing data rate is never larger than the load constraint, U. □

3.2 The Packet Bulk Length: *k*

The major difference between CapProbe and PBProbe is that PBProbe sends packet bulks. The purpose of using packet bulks is to overcome the limited system timer resolution, as well as to avoid the additional latency caused by segmentation and reassembly when the employed packet size is larger than the MTU. Alg. 1 is employed by PBProbe to automatically determine the proper bulk length, k, for capacity estimation.

In the beginning, k is initialized to 1, i.e. PBProbe behaves as CapProbe, using packet-pair to probe the link capacity. However, whenever the measured dispersion is smaller than a certain threshold, say D_{thresh}, this algorithm will increase the bulk length (k) by ten-fold and restart the estimation process. Clearly, the decision of D_{thresh} value depends on the system timer resolution. In this work, we set D_{thresh} = $1ms$ in all experiments.

Corollary 2. *In PBProbe, the difference of the maximum network utilization (i.e. U) and the achieved network utilization (i.e. U') increases as the the bulk length (i.e. k) increases.*

Proof. Let R denotes the introduced data rate of the packet bulk probing, the achieved network utilization is then given by $U' = \frac{R}{C} = \frac{(k+1)P}{CG}$. Since $G = \frac{2D_{m'}}{U}$, $U' = \frac{(k+1)PU}{2CD_{m'}}$.

Let $\triangle U$ denotes the difference of U and U', i.e. $\triangle U = U - U' > 0$, $\triangle U$ is given by:

Algorithm 1 The algorithm for determining the appropriate bulk length, k, in PBProbe.

```
k ← 1
count ← 0
D ← ∞
repeat
  Send START packet
  Receive a packet bulk (of lengh k) and measure D'
  if D' < D_thresh then
    k ← k × 10
    count ← 0
  else
    D ← min(D', D)
    G ← 2D/U
    count ← count + 1
    Sleep(G)
  end if
until count == n
```

$$\triangle U = U - \frac{(k+1)PU}{2CD_{m'}} = \left(\frac{2CD_{m'} - (k+1)P}{2CD_{m'}}\right) U \tag{9}$$

Since $D_{m'} = \frac{kP}{C}$, one can get:

$$\triangle U = \left(\frac{2kP - (k+1)P}{2kP}\right) U = \left(\frac{1}{2} - \frac{1}{2k}\right) U \tag{10}$$

Therefore, $\triangle U$ increases as k increases. □

3.3 The Number of Samples: *n*

CapProbe employs a sophisticated convergence test to determine whether a good sample has been obtained. To simplify the implementation, PBProbe simply estimates link capacity using a fixed number of n samples. Obviously, the larger n is, the more accurately PBProbe estimates. The required time for one PBProbe capacity estimate is linearly proportional to the value of n. More specifically, the larger n is, the longer time PBProbe requires to converge to a given accuracy. Based on the experimental results reported in the previous CapProbe studies [10] [16], we decide to set $n = 200$ throughout this paper.

4 Analysis

In this section, we present a queueing model that can predict the probability of obtaining a "good" sample for a single link with Poisson distributed cross traffic. For simplicity, we assume the probing samples of PBProbe arrive according to a Poisson distribution so that they take so to speak "a random look" at the link. We also assume that the probing samples do not constitute a significant load on the network since they are sent infrequently. Finally, we assume the buffers are large enough so that probing packets will not be dropped due to buffer overflow. The analytical model is described next, followed by the results.

Suppose the arrival and service rate of Poisson cross traffic are λ and μ, the service time of one single probing packet is τ, and the bottleneck link utilization is ρ. The probability of the first probing packet arriving to an empty system, i.e. p, is given by:

$$p = 1 - \frac{\lambda}{\mu} = 1 - \rho \tag{11}$$

Since there is no queueing delay experienced by any probing packets of a "good" sample, the probability of no queueing delay for the remaining k probing packets (i.e. no cross traffic packets arrive in the $k\tau$ period) is $e^{-\lambda(k\tau)}$. Therefore, the probability of obtaining a "good" sample, i.e. p_0, is given by:

$$p_0 = pe^{\lambda(-k\tau)} = (1-\rho)\, e^{-k\lambda\tau} \tag{12}$$

The expected number of samples, $\overline{N}$, for obtaining a good sample is then derived as:

$$\overline{N} = \sum_{n=1}^{\infty} np_0(1-p_0)^{n-1} = \frac{1}{p_0} = \frac{e^{k\lambda\tau}}{1-\rho} \tag{13}$$

Suppose the size of the probing packets and the cross traffic packets are equal, then $\tau = \frac{1}{\mu} = \frac{\rho}{\lambda}$. Therefore, $\overline{N}$ could be rewritten as:

$$\overline{N} = \frac{e^{k\rho}}{1-\rho} \tag{14}$$

The relationship between the expected number of required samples for one good sample ($\overline{N}$) and link utilization (ρ) with different packet bulk length (k) is shown in Fig. 3.

From the results of Fig. 3, when $k = 1$ (i.e. packet-pair based CapProbe), $\overline{N}$ is around 25 while the utilization (ρ) is as high as 0.9. However, as k increases, $\overline{N}$ increases exponentially. For instance, when $\rho = 0.3$[1], $\overline{N}$ is around 30 while $k = 10$, but becomes around 5,000,000 while $k = 50$. It turns out that the estimation speed of PBProbe (i.e. the required number of samples for obtaining a good sample) is highly related to the employed packet bulk length. Though employing a large packet bulk can improve the accuracy of dispersion measurements, such large bulk will need a much larger number of tries in order to lead to a good sample and therefore will slow down the estimation considerably.

[1]The utilization of Abilene backbone network, which is the gigabits backbone of Internet2, is hardly over 30% [1].

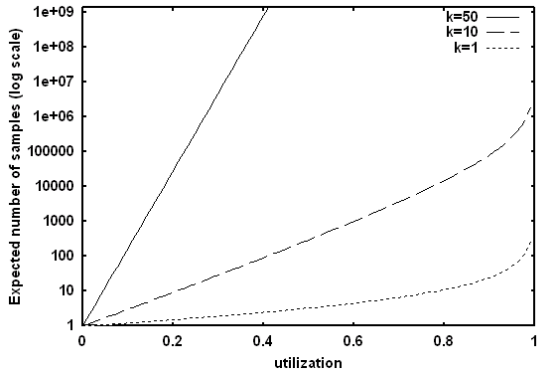

Figure 3: The expected number of samples ($\overline{N}$) with different link utilization (ρ) and packet bulk length (k) under Poisson cross traffic.

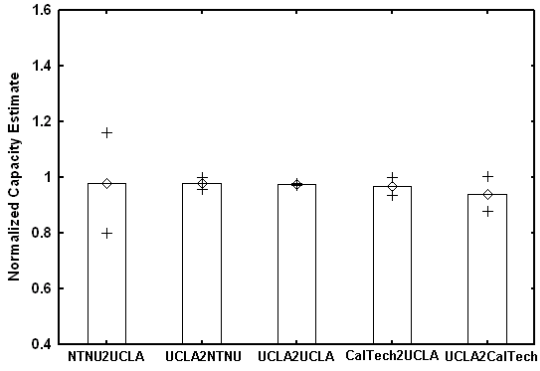

Figure 4: PBProbe capacity estimation (mean and 95% confidence intervals)on 100 Mbps Internet links.

5 Internet Experiments: General Links

In this section, we present experimental results evaluating the accuracy of PBProbe on general Internet connections. We first evaluate PBProbe on the symmetric Internet links in Section 5.1, and then present the results of PBProbe on the emerging first/last-mile asymmetric access links in Sections 5.2 and 5.3.

5.1 Symmetric Links

We first performed Internet experiments on the general symmetric fast Ethernet links. One of the selected paths is a local connection within UCLA campus networks (UCLA $\leftrightarrow$ UCLA; Capacity: 100 Mbps; Round trip time: 0.2 ms), and the other two paths are from UCLA to California Institute of Technology (UCLA $\leftrightarrow$ CalTech; Capacity: 100 Mbps; Round trip time: 8 ms) and from UCLA to National Taiwan Normal University (UCLA $\leftrightarrow$ NTNU; Capacity: 100 Mbps; Round trip time: 180 ms).

For each network path, 20 runs of PBProbe were performed in order to collect the average capacity estimate and standard deviation. The normalized capacity estimates (i.e., the ratio of estimated capacity to real capacity) and 95% confidence intervals are illustrated in Fig. 4. The employed bulk length k and the average time spent for one estimate (i.e. for one direction) of each path are shown in Table 1.

Table 1 shows that PBProbe adapted its bulk length, k, to 10 in all experiments. This is due to the fact that, while $k = 1$ (i.e. PBProbe sent packet pairs as probing samples, which is the same as CapProbe), the dispersion measurements were smaller than the threshold value (i.e. $D_{thresh} = 1ms$). Therefore, the bulk length adaptation algorithm increased k in order to keep measured dispersion larger than D_{thresh} and overcome the system's inadequate timer resolution.

Table 1: Path properties of the employed symmetric Internet connections.

Path	Bulk Length k	Time Spent
UCLA $\leftrightarrow$ NTNU	10	10 sec
UCLA $\leftrightarrow$ CalTech	10	10 sec
UCLA $\leftrightarrow$ UCLA	10	10 sec

Compared with the experiment results presented in [16], PBProbe achieved the same estimation accuracy as CapProbe and Pathrate did on 100 Mbps links. As depicted in Figure 4, PBProbe estimates the bottleneck capacity accurately in most cases. The capacity estimate for the path from NTNU to UCLA fluctuates slightly more than other paths, because the heavy cross traffic on this path reduces the probability of obtaining a good packet bulk sample.

In terms of estimation speed, the performance of PBProbe lies between CapProbe and Pathrate. The reason that PBProbe requires slightly longer time than CapProbe is mainly due to the packet bulk length adaptation and the fixed number of samples. Nevertheless, it is worth noting that, by employing packet bulks instead of packet pairs, PBProbe can adapt to lower timer resolution and higher capacity environments in a more flexible way than CapProbe.

5.2 Asymmetric Links

The emerging first/last-mile Internet access links are usually asymmetric (e.g. DSL and Cable modem links). Previous asymmetric estimation techniques are either very complex (e.g. ALBP [22]) or they require fine-scale system timer resolution support (e.g. AsymProbe [12]). In

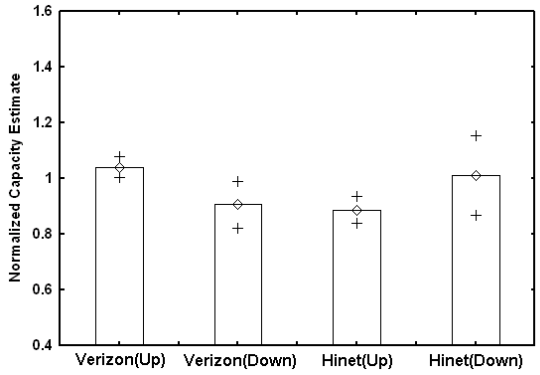

Figure 5: PBProbe experiment results (mean and 95% confidence intervalsof 20 runs) on Up and Down links of DSL connections.

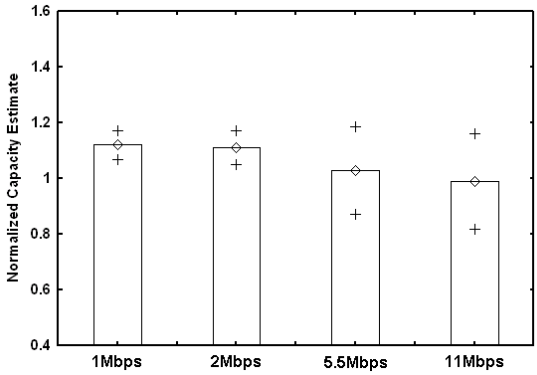

Figure 6: PBProbe experiment results (mean and 95% confidence intervalsof 20 runs) on last mile IEEE 802.11b link with different transmission modes.

this subsection, we performed a set of experiments to evaluate the applicability of PBProbe to asymmetric access links.

Two asymmetric links were selected from our local host (connected to the Internet via a 100 Mbps ethernet link) to two DSL hosts[2]. One of them is Verizon DSL link with 1.5M/384Kbps link capacities on the DOWN/UP links and around 30 ms round-trip delay time. The other is HiNet DSL link with 3M/640Kbps link capacities on the DOWN/UP links and around 50 ms round-trip delay time. The experiments were performed 20 times for each destination host, and k was observed to be 1 in all runs (i.e., when $k = 1$, the measured dispersion was already larger than the threshold value). We plot in Fig. 5 the normalized average capacity estimates with 95% confidence intervals for each direction.

The results clearly demonstrate that PBProbe is able to correctly estimate capacities on all asymmetric links. Specifically, the average capacity estimates are all within 90% accuracy of the real physical link capacity. Given 95% confidence level, all the capacity estimates remains within 80% accuracy.

5.3 Wireless Experiments

Here, we evaluate PBProbe on an one-hop wireless link, which is prevalent in the first/last mile scenarios. The wireless link is IEEE 802.11b based, and the transmission rate is configured to be 1, 2, 5.5, and 11 Mbps respectively. For each transmission rate, we repeated the PBProbe experiment for 20 times in order to collect the average and standard deviation results. Fig. 6 shows the experiment results[3].

From Fig. 6, we observe that PBProbe gives accurate estimates for wireless links as well. The normalized capacity estimates are very close to 1 in all transmission modes, and the standard deviations are small too. It turns out that PBProbe is also able to accurately estimate the capacity of wireless links.

6 Experiments: High-Speed Links

In this section, we evaluate the performance of PBProbe on high speed (i.e., over 100 Mbps) links. We first perform emulation experiments to test the speed and accuracy of PBProbe in a controlled environment. We then conduct Internet experiments to study the performance of PBProbe in a more realistic environment. The experiments results are presented in the following subsections.

6.1 Emulation Experiments

The emulator-based experiments were run on our laboratory testbed with the configurations illustrated in Fig. 7. In the first scenario (i.e. Fig. 7-a), three testing machines are connected serially with 1 Gbps links. The NISTNet emulator [6] is installed on the middle machine and would wisely configure the middle machine to create bottleneck link on the gigabits path. No cross traffic is injected into this path. In the second scenario (i.e. Fig. 7-b), three testing machines are connected to one gigabits switch with 1 Gbps link. A Poisson traffic generator [8] is installed on one of the three machines, and PBProbe was installed on the other two machines. The Poisson

[2]The DSL connections are provided by Verizon (http://www.verizon.net) and HiNet (http://www.hinet.net).

[3]The capacity estimates have been normalized to the effective capacity as reported in http://www.uninett.no/wlan/throughput.html

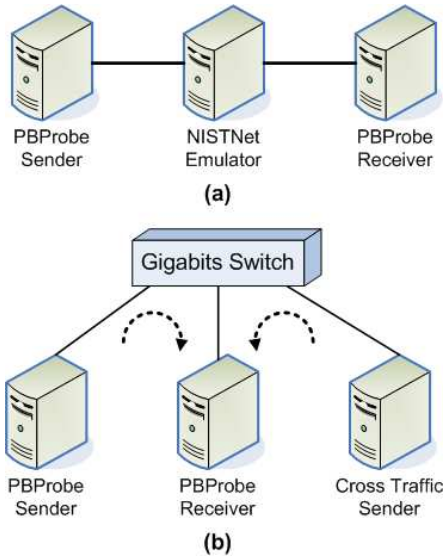

Figure 7: NIST Net emulation testbed

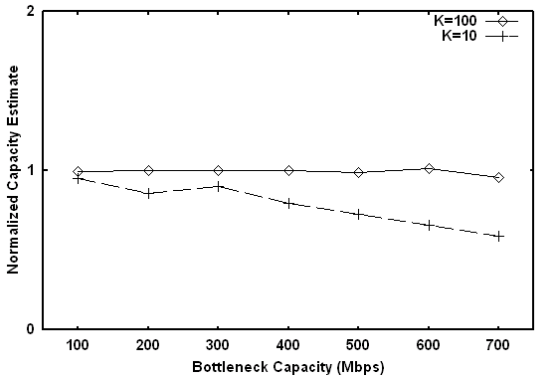

Figure 8: PBProbe estimation results using NISTNet emulator: normalized capacity estimates with various bottleneck capacity settings.

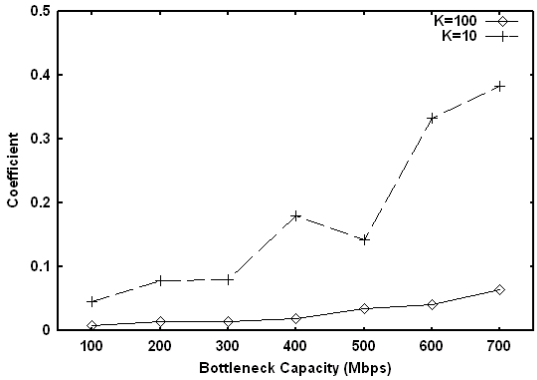

Figure 9: PBProbe estimation results using NISTNet emulator: coefficient of variance of capacity estimates with various bottleneck capacity settings.

traffic generator was configured to generate different levels of cross traffic in order to validate the accuracy and speed of PBProbe under different link utilization levels.

6.1.1 No Cross Traffic

First, we evaluate the accuracy of PBProbe with different k settings on high speed links by disabling the bulk length (k) adaptation algorithm. Fig. 7-a illustrates the testbed configuration, in which no cross traffic was present during the experiments. We varied the bottleneck link capacity, and conducted 20 runs of the experiments for each capacity setting. The results of $k = 10$ and 100 are shown in Fig. 8 and 9, respectively.

Since no cross traffic was present in these experiments, packets within a packet bulk are expected to traverse the path back-to-back without being disturbed. Therefore, ideally, the measured dispersion of each packet bulk should represent the undistorted dispersion corresponding to bottleneck link capacity. Moreover, based on these perfect dispersions, the capacity estimate should be accurate and consistent for every probing sample.

However, since PBProbe requires accurate dispersion measurements, the capacity estimates tend to become inaccurate when timer resolution is inadequate. Table 2 shows the required timer resolution of PBProbe on links with different capacity and for different bulk lengths. Obviously, when the bottleneck link capacity is high or the bulk length is small, a high timer resolution is required. For instance, with link capacity = 100 Mbps and packet pairs (i.e., $k = 1$), only a powerful processor can satisfy the required resolution of 0.12 ms. As the capacity increases to 1 Gbps, the required 0.012 ms resolution becomes very difficult to achieve, and the measurement accuracy will degrade. Indeed, this trend was observed in Fig. 8 and 9. In these two figures, for small k, the capacity estimates become increasingly inaccurate and inconsistent when the required timer resolution became greater (or equivalently, when the bottleneck capacity was enlarged).

However, the results also reveal that, when measuring high capacity links, a large k can successfully alleviate the inaccuracy and inconsistency of capacity estimates caused by poor timer resolution. For instance, when the bottleneck link capacity is 600 Mbps, PBProbe with bulk length $k = 100$ can accurately estimate the capacity (as illustrated by very small coefficient of variance on capacity estimates). In contrast, when $k = 10$, PBProbe can only measure around 60% of the link capacity, and the coefficient of variance is much larger. Based on the ex-

Table 2: Required system timer resolution for accurate PBProbe estimation (assuming probing packet size is 1500 bytes).

k	Bottleneck Link Capacity		
	1Gbps	100Mbps	10Mbps
1	0.012ms	0.12ms	1.2ms
10	0.12ms	1.2ms	12ms
100	1.2ms	12ms	120ms

perimental results as well as the analysis shown in Table 2, we conjecture that our testbed hosts can only provide timer resolution of approximately 1 ms. Therefore, we decided to fix $T_{thresh} = 1ms$ in the k adaptation algorithm of PBProbe for all the following experiments.

6.1.2 Poisson Cross Traffic

The second set of experiments was performed to investigate the impact of different levels of cross traffic on the accuracy and speed of PBProbe. Fig. 7-b) illustrates the testbed topology, in which a Poisson traffic generator injects traffic into the probing path of PBProbe. The Poisson traffic generator varied its traffic rate from 0 to 500 Mbps, and 20 trials of the PBProbe experiment were performed for each cross traffic rate. The bulk length k was fixed to 100. The relationship between the number of employed samples and the average capacity estimates is plotted in Fig. 10.

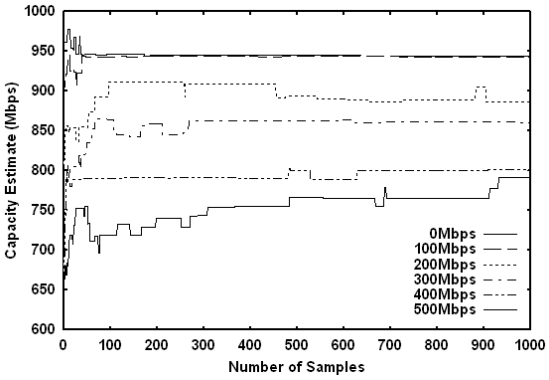

Figure 10: Capacity estimates of PBProbe ($k = 100$) under different Poisson cross traffic.

The results show that PBProbe estimated 950 Mbps capacity very fast (in less than 50 samples) when the Poisson cross traffic was very light (i.e. the Poisson cross traffic varies from 0 to 100 Mbps rate). As the cross traffic increased, the accuracy of PBProbe estimates decreased. Specifically, PBProbe achieved around 90% accuracy when the link utilization was 0.2, 85% accuracy when the link utilization was 0.3. When the link utilization became larger than 40%, PBProbe only achieved around 80% accuracy after 1000 samples. The results are consistent with the analytical results shown in Fig. 3, where the required number of samples increases exponentially with the link utilization.

It turns out that, when the bulk length is large, PBProbe can estimate link capacity very rapidly when the link utilization is low, but it requires a large number of probing samples when the link is highly utilized. Fortunately, most Internet gigabit backbones nowadays are moderately loaded. As reported in [1, 2, 4, 5, 9][4], the utilization of Internet backbones are mostly below 15%. Therefore, in practice, PBProbe is still adequate for capacity estimation on high speed Internet links. To validate the feasibility and capability of PBProbe in more realistic scenarios, we perform experiments on high speed Internet paths in the next subsection.

6.2 Internet Experiments

Here, we conducted Internet experiments to evaluate PBProbe in realistic environments. Five Internet hosts and five Internet gigabits paths (within California: UCLA and CalTech); across country: UCLA - PSC, PSC - GaTech, GaTech - UCLA; and international: NTNU - UCLA) were selected for the experiments. The selected Internet hosts are listed in Table 3, and the topology and path properties of the selected paths are illustrated in Fig. 11.

Table 3: Description of the participating hosts in gigabits Internet experiments.

Abbrev.	Full Name	Location
CalTech	California Institute of Technology	California, USA
GaTech	Georgia Institute of Technology	Georgia, USA
NTNU	National Taiwan Normal University	Taipei, Taiwan
PSC	Pittsburgh Supercomputing Center	Pennsylvania, USA
UCLA	University of California at Los Angeles	California, USA

Fig. 12 and 13 illustrate the experiment results (i.e. normalized mean and coefficient of variance of capacity estimates in 20 runs). It is clear that, in Fig. 12, the normalized capacity estimates are mostly within 90% accuracy range, except three outgoing links from UCLA to CalTech, GaTech, and PSC. This is due to the fact that the outgoing link of UCLA backbone has around 30% utilization, which is much higher than the utilization of the incoming link (around 15%) [2], and, in this case,

[4]More information of Internet utilization can be found at http://netmon.grnet.gr/weathermap/ and http://people.ee.ethz.ch/ oetiker/webtools/mrtg/users.html

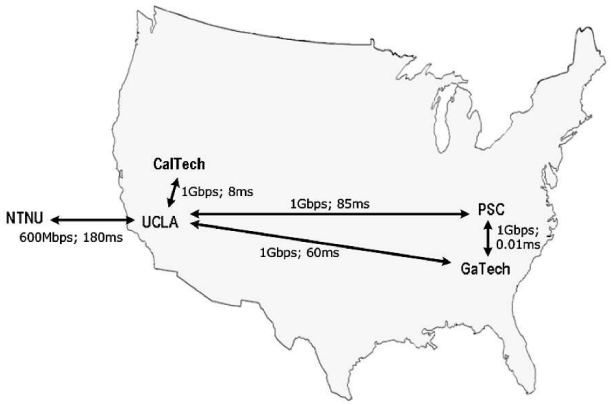

Figure 11: Topology and path properties (bottleneck capacity and round trip delay) of selected gigabits Internet paths.

Table 4: Comparison of PBProbe and Pathrate on Internet gigabits links. (Capacity: Mbps; Time: seconds)

	PBProbe		Pathrate	
	Capacity	Time	Capacity	Time
CalTech → UCLA (1Gbps)	919.6	14	933.2	17
UCLA → CalTech (1Gbps)	839.4	14	945.3	1146
GaTech → UCLA (1Gbps)	928.1	14	932.9	18
UCLA → GaTech (1Gbps)	840.3	14	968.1	1223
PSC → GaTech (1Gbps)	959.9	13	995.4	1122
GaTech → PSC (1Gbps)	905.9	13	947.0	17
PSC → UCLA (1Gbps)	889.6	14	935.6	20
UCLA → PSC (1Gbps)	845.2	15	905.6	20
NTNU → UCLA (600Mbps)	580.6	20	575.6	1641
UCLA → NTNU (600Mbps)	588.4	21	573.4	1641

PBProbe requires a large number of samples in order to correctly estimate the capacity. Since we set $n = 200$ in all experiments, PBProbe only estimated around 80% of the link capacity, which is consistent with the results presented in subsection 6.1.2.

In addition, Fig. 13 also shows that the coefficient of variance of PBProbe estimates are below 0.15 on all tested links, i.e. the capacity estimates (20 runs) of each high-speed link are very stable. Comparing with the results shown in 9, it turns out that PBProbe was able to adapt its bulk length to the most appropriate value (i.e. $k = 100$) so that it could consistently and accurately estimate the capacities of high speed links.

6.3 Comparisons of PBProbe and Pathrate

So far, we have evaluated PBProbe in a variety of network scenarios. In this subsection, we compare the performance of PBProbe and Pathrate in terms of accuracy, speed, and bandwidth consumption (i.e. overhead). The experiments are performed on the same set of high speed Internet links as illustrated in Fig. 11, and the results of capacity estimates and required time are shown in Table 4.

In Table 4, PBProbe measures at least around 85% bottleneck capacities of all tested links; whereas Pathrate is very accurate and measures at least 90% bottleneck capacities for all links. However, for some links with long delay and/or high utilization, Pathrate required more than 1000 seconds to estimate the capacity. This is due to the fact that, if the distribution of measured dispersion is not unimodal after the first phase, Pathrate will start the second phase to probe the network using different packet train lengths and packet sizes. Once Pathrate enters the second phase, it takes long time to determine the correct link capacity. As a result, Pathrate converges fast if the dispersion distribution is unimodal in the first phase, but it becomes much slower than PBProbe otherwise.

We also compared the packet overhead caused by PBProbe and Pathrate on high speed Internet links. Table 5 shows the comparison on one of the high speed links, the UCLA - GaTech link. From the experiment results, PBProbe is more expensive than Pathrate, if Pathrate only uses one phase to estimate link capacity on high speed links. In case when Pathrate is required to enter the second phase, PBProbe and Pathrate produce a comparable amount of packet overhead. However, since Pathrate is much slower after entering the second phase, the bandwidth consumption (i.e. bits per second) is much smaller than PBProbe. It is worth pointing out that, even though the bandwidth consumption of PBProbe (approximately 2 Mbps) seems to be relatively high, it is only 0.2% of the bottleneck capacity (i.e., 1 Gbps); thus, it is not intrusive to other traffic flows in the network.

The experiment results suggest that there are trade-offs between PBProbe and Pathrate for high-speed path capacity estimation. On the one hand, PBProbe yields good estimation results very fast (e.g., less than 20 seconds in most cases). If given more time, it will progressively improve the estimates, since better samples can be obtained. On the other hand, Pathrate tends to produce accurate results, but the required time may vary from approximately 20 seconds to 20 minutes. Thus, Pathrate is not indicated in scenarios when an estimation of the bottleneck capacity needs to be obtained within a very short time.

It should also be noted that the packet overhead of PBProbe is proportional to the employed bulk length k. While measuring a high speed link, PBProbe increases its bulk length and in turn increases the packet overhead, in order to overcome the limited support of system timer resolution. Nonetheless, thank to the employed U pa-

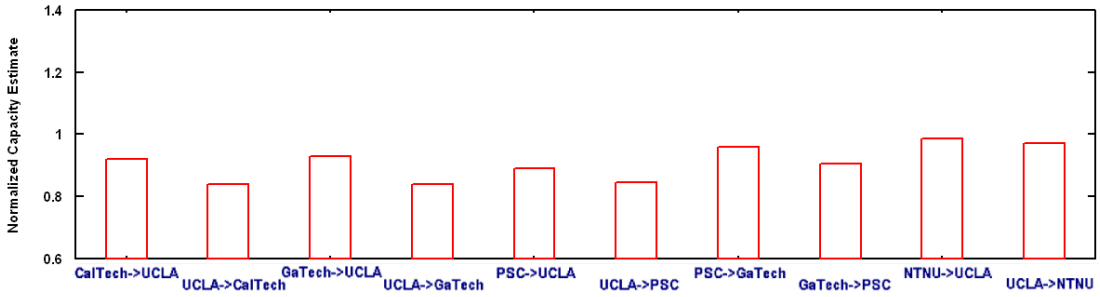

Figure 12: PBProbe experiment results (mean of 20 runs) on high speed links.

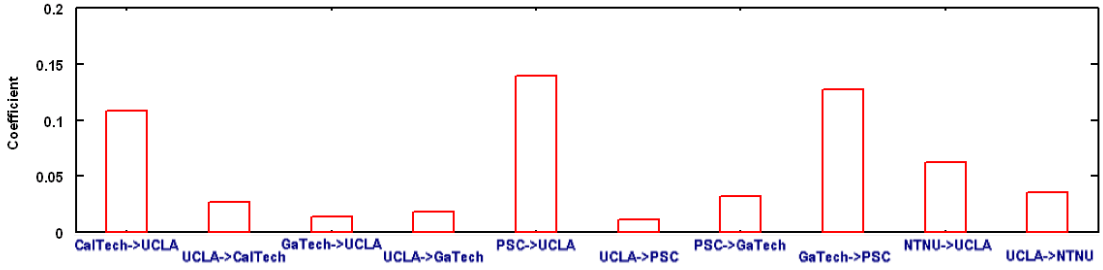

Figure 13: PBProbe experiment results (coefficient of variance of 20 runs) on high speed links.

Table 5: Comparison of PBProbe and Pathrate overhead on Internet gigabits links.

	GaTech → UCLA		UCLA → GaTech	
	PBProbe	Pathrate	PBProbe	Pathrate
spent time	14 sec	18 sec	14 sec	1223 sec
total packets	20,213	2,414	20,213	27,630
total bytes	30,319,500	3,543,752	30,319,500	39,707,740
BW consumption	2.166Mbps	1.575Mbps	2.166Mbps	0.260Mbps

rameter, the bandwidth consumption of PBProbe is restricted by the utilization upper bound. Hence, PBProbe can carefully control the trade-off between the bandwidth consumption and the required time in order to satisfy the requirement of different applications.

To summarize, PBProbe is scalable and adaptive to network link capacities, and applicable to asymmetric access links and one-hop wireless links. It is consistently fast, accurate, and non-intrusive. Therefore, PBProbe is indeed feasible to be deployed and applied in the emerging network applications.

7 Conclusions

In this paper, we studied a classic problem of link capacity estimation, and we proposed a scalable technique, called PBProbe, to estimate bottleneck capacity of a path with a variety of network dynamics. PBProbe is based on the CapProbe algorithm, and it employs "packet bulk" technique to adapt the number of packets in each probing in accordance to the network characteristics. As a result, it preserves the simplicity, speed, and accuracy of CapProbe in estimating general Internet paths, and it overcomes the poor system timer resolution support on high speed links. Using analysis, emulation, and Internet experiments, we evaluated the accuracy, speed, and overhead of PBProbe on various network configurations. The results show that PBProbe is able to correctly and fast estimate bottleneck capacity in almost all test cases. PBProbe is ideal in real deployments where online and timely capacity estimation is required. Typical applications are peer-to-peer streaming and file sharing, overlay network structuring, and network monitoring, pricing and QoS enhancements.

8 Acknowledgments

We are grateful to the following people for their help in carrying out PBProbe measurements: Yu-Chin Fang (National Taiwan Normal University), Sanjay Hegde (California Institute of Technology), Che-Chih Liu (National Taiwan Normal University), Cesar A. C. Marcondes (University of California, Los Angeles), and Anders Persson (University of California, Los Angeles).

References

[1] Abilene network traffic. http://loadrunner.uits.iu.edu/weathermaps/abilene/.

[2] Cenic network statistics. http://cricket.cenic.org/grapher.cgi.

[3] Clink: a tool for estimating internet link characteristics. http://allendowney.com/research/clink/.

[4] Garr-b network weather map. http://www.noc.garr.it/mappe/backbone.shtml.

[5] Grnet: Network weathermap. http://netmon.grnet.gr/map.shtml.

[6] Nistnet: network emulation package. http://www.antd.nist.gov/itg/nistnet/.

[7] pchar: A tool for measuring internet path characteristics. http://www.kitchenlab.org/www/bmah/Software/pchar/.

[8] Poisson traffic generator. http://www.spin.rice.edu/Software/poisson_gen/.

[9] Twaren weather map. http://mrtg.twaren.net/mrtg/wmap/.

[10] CHEN, L.-J., SUN, T., XU, D., SANADIDI, M. Y., AND GERLA, M. Access link capacity monitoring with tfrc probe. In *E2EMON* (2004).

[11] CHEN, L.-J., SUN, T., YANG, G., SANADIDI, M. Y., AND GERLA, M. Adhoc probe: Path capacity probing in ad hoc networks. In *WICON* (2005).

[12] CHEN, L.-J., SUN, T., YANG, G., SANADIDI, M. Y., AND GERLA, M. End-to-end asymmetric link capacity estimation. In *IFIP Networking* (2005).

[13] DOVROLIS, C., RAMANATHAN, P., AND MOORE, D. What do packet dispersion techniques measure? In *IEEE Infocom* (2001).

[14] JACOBSON, V. Pathchar: A tool to infer characteristics of internet paths. ftp://ftp.ee.lbl.gov/pathchar/.

[15] JIN, G., AND TIERNEY, B. System capability effect on algorithms for network bandwidth measurement. In *ACM IMC* (2003).

[16] KAPOOR, R., CHEN, L.-J., LAO, L., GERLA, M., AND SANADIDI, M. Y. Capprobe: A simple and accurate capacity estimation technique. In *ACM SIGCOMM* (2004).

[17] KAPOOR, R., CHEN, L.-J., SANADIDI, M. Y., AND GERLA, M. Accuracy of link capacity estimates using passive and active approaches with capprobe. In *IEEE ISCC* (2004).

[18] LAI, K., AND BAKER, M. Measuring bandwidth. In *IEEE Infocom* (1999), pp. 235–245.

[19] LAI, K., AND BAKER, M. Measuring link bandwidths using a deterministic model of packet delay. In *ACM SIGCOMM* (2000).

[20] LAKSHMINARAYANAN, K., PADMANABHAN, V. N., AND PADHYE, J. Bandwidth estimation in broadband access networks. In *IMC* (2004).

[21] LEE, S.-J., SHARMA, P., BANERJEE, S., BASU, S., AND FONSECA, R. Measuring bandwidth between planetlab nodes. In *PAM* (2005).

[22] LIN, Y., WU, H., CHENG, S., WANG, W., AND WANG, C. Measuring asymmetric link bandwidths in internet using a multipacket delay model. In *IEEE ICC* (2003).

[23] POSTEL, J. Internet protocol. Tech. rep., IETF RFC 791, September 1981.

[24] PRASAD, R., JAIN, M., AND DOVROLIS, C. Evaluating pathrate and pathload with realistic cross-traffic. http://www.cc.gatech.edu/ jain/pub/talk/best03_talk.ppt, 2003 Bandwidth Estimation Workshop.

[25] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. D. Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments. In *IEEE Infocom* (2002).

[26] SAVAGE, S. Sting: a tcp-based network measurement tool. In *USENIX Symposium on Internet Technologies and Systems* (1999).