Estimating Link Capacity in High Speed Networks⋆

Ling-Jyh Chen[1], Tony Sun[2], Li Lao[2], Guang Yang[2], M. Y. Sanadidi[2], Mario Gerla[2]

[1] Institute of Information Science, Academia Sinica, Taipei 11529, Taiwan
[2] Department of Computer Science, UCLA, Los Angeles, CA 90095, USA

Abstract. Knowledge of bottleneck capacity of an Internet path is critical for efficient network design, management, and usage. With emerging high speed Internet links, most traditional estimation techniques are limited in providing fast and accurate capacity estimations. In this paper, we propose a new technique, called PBProbe, to estimate high speed links. PBProbe is based on CapProbe; however, instead of solely relying on packet pairs, PBProbe employs a "packet bulk" technique and adapts the bulk length in order to overcome the 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 PBProbe with various bulk lengths and network configurations. We then perform emulation and Internet experiments to verify the accuracy and speed of PBProbe on 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. To accommodate the diversity in network arrangements, an ideal capacity estimation tool must also be scalable and applicable to a variety of network configurations.

A number of techniques have been proposed for capacity estimation on generic Internet paths [1–7]. Among them, CapProbe [5] and Pathrate [3] 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 capacities of asymmetric links [8], and Pathrate performs poorly on wireless links [5].

To address the problems above, specialized capacity estimation tools have been proposed for specific and emerging network scenarios. For instance, ALBP [9] and

---

AsymProbe [8] are intended for capacity estimation on asymmetric links, and AdHoc Probe [10] aims to estimate the end-to-end path capacity in wireless networks. However, for emerging high speed network links (i.e., gigabit links), recent studies have showed the standing challenging to estimate high speed link capacity (various system issues) [11], and a simple, fast and accurate technique is still lacking and desirable.

In this paper, we propose a capacity estimation tool for high speed network links, called PBProbe. PBProbe is inspired by CapProbe. However, instead of solely 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, which enlarges the dispersion between the first and and last packet, to overcome the known timer resolution problem. As we will discuss in more detail later in the paper, timer resolution is the main challenge in estimating high capacity link capacities [11, 12].

The rest of the paper is organized as follows. In section 2, we 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 evaluate PBProbe on high speed links in our emulator testbed as well as on the Internet. Section 6 concludes the paper.

2   Related Work

Previous research on capacity estimation relied either on delay variations among probe packets as illustrated in Pathchar [4], or on dispersion among probe packets as described in Nettimer [6] and Pathrate [3]. Pathchar-like tools (such as pchar [2] and clink [1]) have limitations in accuracy and speed as shown in [5] [13]. Moreover, they evaluate the capacity of a link based on the estimates of previous links along the path, thus estimation errors accumulate and amplify with each measured link [7].

Dispersion-based techniques suffer from other problems. In particular, Dovrolis' analysis in [3] 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) the Average Dispersion Rate (ADR), which is always lower than capacity. Another dispersion-based tool, SProbe [7], exploits SYN and RST packets of the TCP protocol to estimate the downstream link capacity, and employs two heuristics to filter out samples which have experienced cross traffic. However, SProbe does not work properly when the network is highly utilized [14].

Unlike the above approaches, CapProbe [5] uses both dispersion measurements and end-to-end delay measurements to filter out the packet pair samples that were 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 does not reflect the higher capacity link 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) [15].

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 [16]. Chen et al. extended CapProbe to estimate *effective path capacity* in ad hoc wireless networks [10]. In addition, ABLP [9] and AsymProbe [8] have been proposed for capacity estimation on the increasingly popular asymmetric links (e.g. DSL and satellite links).

Nonetheless, capacity estimation on high speed links remains a challenge. Though recent studies have verified the accuracy of Pathrate in estimating gigabit links [17], however, the evaluation was done on an emulator-based testbed, which cannot represent realistic Internet dynamics. Thus, an experimental evaluation of capacity estimation on high speed links is still lacking.

In this paper, we propose a novel packet bulk technique for estimating high speed link capacities, called PBProbe. PBProbe is based on CapProbe, but it probes the bottleneck link capacity using UDP packets (instead of the ICMP packets used by CapProbe). We present the PBProbe algorithm in the next section.

## 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. 1 illustrates the algorithm of PBProbe.

In the first phase (as shown in Fig. 1-a), host A first sends a *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:

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

$$D_i = t_{rcv}(i, k + 1) - t_{rcv}(i, 1) \tag{2}$$

If none of the $k + 1$ probing packets experience cross-traffic induced queueing, the sample will reflect the correct capacity. Thus, the "good" sample (say, the m-th sample)
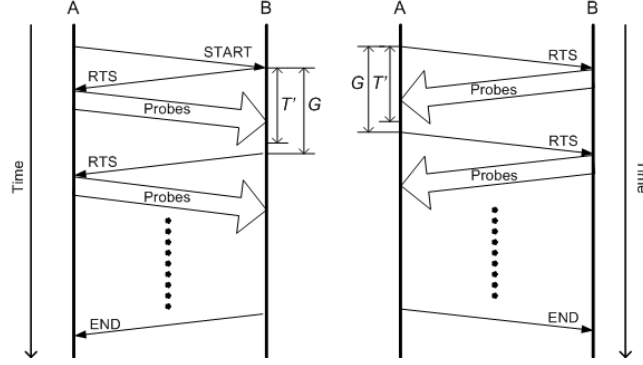
Fig. 1. Illustration of PBProbe (a) Phase I: measuring forward direction link capacity; (b) Phase II: measuring backward direction link capacity.

is identified by applying the minimum delay sum filter to all probing samples (say, n samples):

$$m = arg \min_{i=1...n} S_i \qquad (3)$$

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

$$C = \frac{kP}{D_m} \qquad (4)$$

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. 1-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 networks and thus requires longer time to converge. Therefore, for PBProbe, we set G to be:

$$G = \frac{2D_{m'}}{U} \qquad (5)$$

```
k ← 1 ; count ← 0 ; D ← ∞
repeat
    t₁ ← time()
    Send START packet
    Receive a packet bulk (of length 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 ;  t₂ ← time()
        Sleep(G - (t₂ - t₁))
    end if
until count == n
```

Fig. 2. The algorithm for determining the appropriate bulk length, k, in PBProbe.

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. We provide a short proof below showing that *if $G = \frac{2D_{m'}}{U}$, the network utilization constraint, U, can be guaranteed.*

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

$$G = \frac{2D_{m'}}{U} \geq \frac{D_{m'}}{U} + \frac{D_{m'}}{kU} = \frac{kP}{CU} + \frac{P}{CU} = \frac{(k+1)P}{CU} \quad (6)$$

$$\Rightarrow \frac{(k+1)P}{G} \leq CU \quad (7)$$

Let R denote the data rate of the introduced packet bulk probes , 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 packet size used is larger than the MTU. The algorithm in Fig. 2 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 like 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 $D_{thresh} = 1ms$ for all the experiments

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 capacities 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. Based on the experimental results reported in the previous CapProbe studies [18] [5], we decide to set $n = 200$ throughout this paper.

4 Analysis

In this section, we present a queueing model that predicts 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 process 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:

$$p = 1 - \frac{\lambda}{\mu} = 1 - \rho \tag{8}$$

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:

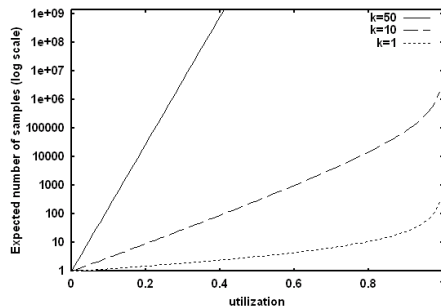$$p_0 = pe^{\lambda(-k\tau)} = (1 - \rho)\, e^{-k\lambda\tau} \tag{9}$$

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

$$\overline{N} = \sum_{n=1}^{\infty} np_0(1 - p_0)^{n-1} = \frac{1}{p_0} = \frac{e^{k\lambda\tau}}{1 - \rho} \tag{10}$$

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:

$$\overline{N} = \frac{e^{k\rho}}{1 - \rho} \tag{11}$$

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.

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

From the results illustrated in Fig. 3, when $k = 1$ (i.e. packet-pair based CapProbe), $\overline{N}$ is around 25 when the utilization (ρ) is as high as 0.9. However, as k increases, $\overline{N}$ increases exponentially. For instance, when $\rho = 0.3^3$, $\overline{N}$ is around 30 when $k = 10$, but becomes around 5,000,000 when $k = 50$. 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.

5 Evaluation

5.1 Emulation Experiments

The emulator-based experiments were run on our laboratory testbed. In the scenario, three testing machines are connected serially with 1 Gbps links. The NISTNet emulator [20] is installed on the middle machine and can configure the middle machine to create bottleneck link on the gigabit path. The purpose of this set of experiments is to verify the needs of bulk length adaption for estimating high speed links; therefore, we used two fixed bulk lengths (i.e., $k = 10$ and 100) and did not employ cross traffic in the emulation experiments. We varied the bottleneck link capacity, and conducted 30 runs of the experiments for each capacity setting. The results of $k = 10$ and 100 are shown in Fig. 4.

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

---

[3] The utilization of Abilene backbone network, which is the gigabit backbone of Internet2, is hardly over 30% [19].
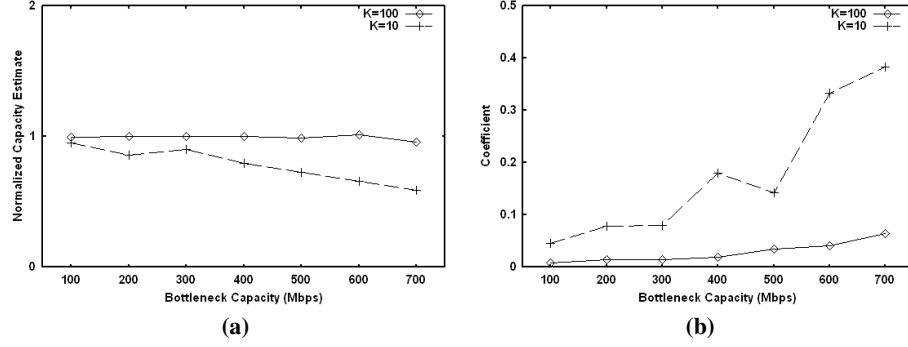
Fig. 4. PBProbe estimation results using NISTNet emulator: (a) normalized capacity estimates and (b) coefficient of variation of capacity estimates with various bottleneck capacity settings.

Table 1. Required system timer resolution for accurate PBProbe estimation (assuming probing packet size is 1500 bytes).
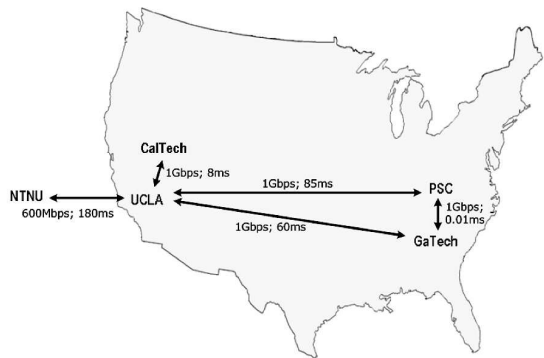
k	Bottleneck Link Capacity		
	1Gbps	100Mbps	10Mbps
1	0.012ms	0.12ms	1.2ms
10	0.12ms	1.2ms	12ms
100	1.2ms	12ms	120ms

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 1 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. 4-a and 4-b. 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 variation on capacity estimates). In contrast, when $k = 10$, PBProbe can only measure around 60% of the link capacity, and the coefficient of variation is much larger. Based on the experimental results as well as the analysis shown

Fig. 5. Topology and path properties (bottleneck capacity and round trip delay) of selected gigabit Internet paths.

in Table 1, 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.
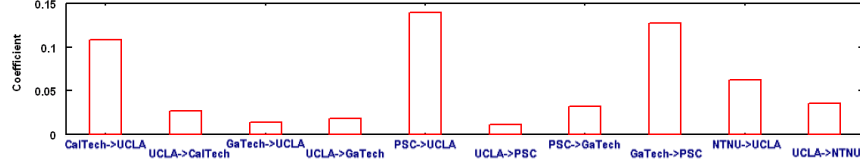
5.2 Internet Experiments

Here, we conducted Internet experiments to evaluate PBProbe in realistic environments. Five Internet hosts (CalTech: California Institute of Technology; GaTech: Georgia Institute of Technology; NTNU: National Taiwan Normal University; PSC: Pittsburgh Supercomputing Center; UCLA: University of California at Los Angeles) and five Internet gigabit paths (within California: UCLA and CalTech); across country: UCLA - PSC, PSC - GaTech, GaTech - UCLA; and international: NTNU - UCLA) were selected for the experiments. The topology and path properties of the selected paths are illustrated in Fig. 5.

Fig. 6 and 7 illustrate the experiment results (i.e. normalized mean and coefficient of variation of capacity estimates in 20 runs). It is clear that, in Fig. 6, 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%) [21], and, in this case, 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.

In addition, Fig. 7 also shows that the coefficient of variation 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 Fig. 4, 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.
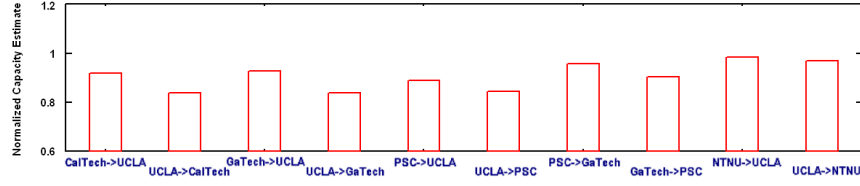
Fig. 6. PBProbe experiment results (mean of 20 runs) on high speed links.

Fig. 7. PBProbe experiment results (coefficient of variation of 20 runs) on high speed links.

Table 2. Comparison of PBProbe and Pathrate on Internet links. (Capacity: Mbps; Time: sec)

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

Table 3. Comparison of PBProbe and Pathrate overhead on Internet gigabit 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

5.3 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. 5, and the results of capacity estimates and required time are shown in Table 2.

In Table 2, 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 3 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 realistically 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 very good estimation results rapidly (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. Therefore, Pathrate may not be ideal 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 parameter, 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.

6   Conclusions

In this paper, we studied a classic problem of link capacity estimation, and we proposed a technique, called PBProbe, to estimate bottleneck capacity for emerging high speed links. PBProbe is based on the CapProbe algorithm, but it uses "packet bulk" to adapt the number of packets in each probing according to different network characteristics. As a result, it preserves the simplicity, speed, and accuracy of CapProbe, as well as overcoming the poor system timer resolution problem 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 can correctly and rapidly estimate bottleneck capacity in almost all test cases. Comparing to other capacity estimation techniques, PBProbe is ideal in real deployments that requires online and timely capacity estimation. This capacity estimation technique can further provide assistance to typical applications such as peer-to-peer streaming and file shar-

ing, overlay network structuring, pricing and QoS enhancements, as well as network monitoring.

7 Acknowledgments

References

1. "Clink: a tool for estimating internet link characteristics," http://allendowney.com/research/clink/.
2. "pchar: A tool for measuring internet path characteristics," http://www.kitchenlab.org/www/bmah/Software/pchar/.
3. C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *IEEE Infocom*, 2001.
4. V. Jacobson, "Pathchar: A tool to infer characteristics of internet paths," ftp://ftp.ee.lbl.gov/pathchar/. [Online]. Available: ftp://ftp.ee.lbl.gov/pathchar/
5. R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi, "Capprobe: A simple and accurate capacity estimation technique," in *ACM SIGCOMM*, 2004.
6. K. Lai and M. Baker, "Measuring bandwidth," in *IEEE Infocom*, 1999, pp. 235–245.
7. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments," in *IEEE Infocom*, 2002.
8. L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, "End-to-end asymmetric link capacity estimation," in *IFIP Networking*, 2005.
9. Y. Lin, H. Wu, S. Cheng, W. Wang, and C. Wang, "Measuring asymmetric link bandwidths in internet using a multi-packet delay model," in *IEEE ICC*, 2003.
10. L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, "Adhoc probe: Path capacity probing in ad hoc networks," in *WICON*, 2005.
11. G. Jin and B. Tierney, "System capability effect on algorithms for network bandwidth measurement," in *ACM IMC*, 2003.
12. R. Kapoor, L.-J. Chen, M. Y. Sanadidi, and M. Gerla, "Accuracy of link capacity estimates using passive and active approaches with capprobe," in *IEEE ISCC*, 2004.
13. K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *ACM SIGCOMM*, 2000.
14. S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," in *PAM*, 2005.
15. S. Savage, "Sting: a tcp-based network measurement tool," in *USENIX Symposium on Internet Technologies and Systems*, 1999.
16. K. Lakshminarayanan, V. N. Padmanabhan, and J. Padhye, "Bandwidth estimation in broadband access networks," in *IMC*, 2004.
17. R. Prasad, M. Jain, and C. Dovrolis, "Evaluating pathrate and pathload with realistic cross-traffic," http://www.cc.gatech.edu/ jain/pub/talk/best03_talk.ppt, 2003 Bandwidth Estimation Workshop. [Online]. Available: http://www.cc.gatech.edu/ jain/pub/talk/best03_talk.ppt
18. L.-J. Chen, T. Sun, D. Xu, M. Y. Sanadidi, and M. Gerla, "Access link capacity monitoring with tfrc probe," in *E2EMON*, 2004.
19. "Abilene network traffic," http://loadrunner.uits.iu.edu/weathermaps/abilene/.
20. "Nistnet: network emulation package," http://www.antd.nist.gov/itg/nistnet/.
21. "Cenic network statistics," http://cricket.cenic.org/grapher.cgi.