On Estimating Tight-Link Bandwidth Characteristics over Multi-Hop Paths

Seong-Ryong Kang*, Xiliang Liu†, Amit Bhati*, and Dmitri Loguinov*
*Department of Computer Science, Texas A&M University, College Station, TX 77843
Email: {skang, abhati, dmitri}@cs.tamu.edu
†City University of New York, New York, NY 10016
Email: xliu@gc.cuny.edu

***Abstract*— In this paper, we explore multi-hop bandwidth estimation assuming arbitrary cross-traffic at each node and develop a new probing method called Envelope that can asymptotically estimate not only the available bandwidth but also the raw capacity of the tight link. Envelope is based on a multi-link recursive extension of unbiased single-hop estimators proposed in the past (e.g., [14]) and a variation of the packet-cartouche technique [6]. Through extensive simulations, we evaluate Envelope in various network settings and cross-traffic conditions and find that it can measure tight-link bandwidth characteristics with accuracy that significantly surpasses that of the existing methods. We also find that Envelope can measure non-tight links in certain path and cross-traffic configurations.**

I. Introduction

Measuring bandwidth characteristics of Internet paths has attracted significant interest over the years. Research in this area has initially focused on the link with the minimum capacity (i.e., the *narrow* link) along the path [1], [2], [3], [4], [6], [16]. More recently, measuring bandwidth characteristics of the link with the minimum available bandwidth (i.e., the *tight* link) received substantially more attention. A number of techniques have been proposed to measure the available bandwidth [9], [12], [24], [26], cross-traffic rate [7], [9], [21], [23], capacity [14], [21], and location [8], [25] of the tight link.

While a variety of methods currently exists, the majority of them are designed based on a single-hop model, which does not consider the cross-traffic interference at non-tight links. This approach imposes two fundamental limitations on the existing techniques. First, most available bandwidth estimation tools do not have a provable convergence to the true value and are often subject to significant measurement biases due to interference in non-tight links [19]. Second, and more importantly, there is no existing technique that can correctly measure the tight-link capacity over a multi-hop path, even though a provably accurate estimator exists for a single-hop path [14]. Note that the tight-link capacity is an important metric in understanding the characteristics of an end-to-end path and several measurement tools (e.g., [9], [26]) require it in order to estimate the available bandwidth of the path.

In this paper, our goal is to measure with *asymptotic accuracy*[1] both capacity and available bandwidth of the tight link over a multi-hop path under arbitrary cross-traffic. We present a recursive extension of the single-node estimator proposed in [14] to multiple routers and combine it with the packet-cartouche technique [6] to sample the statistics of each hop independently. We develop a bandwidth estimation method called *Envelope* that sends trains of probe packets surrounded by two Envelope packets along the path in question. The key idea of this method is to allow the probe packets to sample queuing dynamics at the *desired* router R_k and then "disappear" from the network at router R_{k+1}. By selecting probe parameters to ensure that spacing z_k between the *surviving* Envelope packets at router R_k is sufficiently large, the method guarantees that the mean dispersion of the entire train at router R_k (i.e., $E[z_k]$) is preserved in the path suffix and that the receiver (whenever feasible) is able to extract available bandwidth A_k and capacity C_k from $E[z_k]$.

A complete measurement using Envelope takes $M-1$ phases, where M is the number of links in the end-to-end path. Each phase focuses on a particular router R_k and sends enough probe-trains to obtain the mean spacing $E[z_k]$ exiting from R_k. Armed with this information and the mean spacing $E[z_{k-1}]$ measured in the previous phase, Envelope is able to estimate the available bandwidth A_k and capacity C_k of router R_k given that the condition $E[z_k] > E[z_{k-1}]$ is satisfied. If this condition is not satisfied, router R_k is marked as "unmeasurable." Note that Envelope can select probing parameters such that the tight link is always measurable (for more on this issue, see Section VI-A).

The major advantage of Envelope over existing methods is its asymptotic accuracy in measuring both bandwidth metrics of the tight link. This stems from the fact that Envelope is able to sample uncorrupted inter-packet spacing information at the tight link from the mean dispersion of the probe-train. Note that since Envelope examines each link on a path separately to obtain the uncorrupted information about the tight link, it sometimes requires more probe packets and time than existing available bandwidth estimators. However, we believe that the additional overhead is warranted in cases that the tight-link capacity information (which can not be measured by previous techniques) is highly desirable.

Through `ns2` simulations in various network settings with multiple links, we find that Envelope indeed correctly identifies the tight link of the path and produces the capacity and

[1] Asymptotic accuracy means the convergence of estimates to the true value after a sufficiently long measurement process.

available bandwidth estimates with over 90% accuracy. We compare Envelope with existing available bandwidth estimation tools Pathload [12], Spruce [26], and IGI [9]. We also compare Envelope with existing capacity estimation tools such as Pathrate [4] and CapProbe [15] when the narrow link is measurable by Envelope. Our results show that Envelope exhibits similar performance to that of Pathload in measuring available bandwidth, but significantly outperforms the other existing approaches under all studied conditions.

The rest of the paper is organized as follows. Section II discusses background and related work. Section III introduces Envelope and Section IV evaluates its performance in various network settings. Section V examines several existing methods and their estimation accuracy. Section VI discusses probe parameters used in Envelope and Section VII concludes the paper.

II. Background and Related Work

In this section, we summarize related work in end-to-end capacity and available bandwidth measurement and introduce the single-hop analytical model that Envelope is based upon.

A. Capacity Measurement

The vast majority of previous capacity measurement research focused on estimating the minimum link capacity (i.e., the narrow-link capacity) of a path. The basic idea for all the work in this area stems from an observation by Jacobson [11] in 1988 that two back-to-back packets (a packet-pair) of size s each injected into an empty network path come out with their dispersion equal to s/C_m, where C_m is the minimum link capacity of the path[2]. In real networks, the output packet-pair dispersions are distorted by cross-traffic and exhibit a multi-modal distribution [22]. Hence, subsequent work centered around how to identify in the output dispersion histogram the mode that corresponds to the minimum capacity and how to enhance the capacity mode to facilitate its detection. These efforts led to several measurement methods and public tools such as bprobe [3], PBM [22], Nettimer [16], and Pathrate [4]. These tools work well in certain network environments, but there is in general no guarantee that their capacity estimates converge to the correct value even when a sufficient number of probing samples are used.

In addition to histogram-based proposals, other methods include *packet tailgating* [17], where larger packets are followed by smaller packets to ensure a particular queuing pattern at the narrow link, and *packet cartouche* [6], where certain packets in a probe-train are dropped at select routers (using the TTL field) so as to ensure that the surviving packets can measure the capacity of individual routers and/or subpaths. Additional router-assisted bottleneck bandwidth estimation techniques can be found in [5], [10], [20].

Besides the methods discussed above, a recent approach called CapProbe [15] is based on the idea that if two back-to-back packets are never queued along a given path, then their inter-arrival spacing at the receiver represents the transmission delay of the probe packet at the narrow link and the one-way delay sum of the packets is the minimum among all packet-pairs. Using minimum filtering, CapProbe is frequently able to obtain C_m with better accuracy and much quicker than the previous methods; however, its asymptotic accuracy in congested paths remains unexplored.

[2]We use m to index the narrow link and t to index the tight link.

B. Available Bandwidth Measurement

Unlike a link capacity, available bandwidth is a dynamic metric that varies over time. Most existing work measures the average available bandwidth of the tight link over a long time interval (e.g., tens of seconds). The fundamental idea is based on utilizing the *statistical mean* of packet-train output dispersions and its relationship to the input dispersion. Such relationship was first derived using a single-hop path with constant-rate fluid cross-traffic in [4], which led to a number of proposed techniques that can be classified into two categories [13].

The first category is called *iterative probing*, which includes Pathload [12], PTR [9], and TOPP [21]. These techniques iteratively adjust input rates of packet-trains to identify the transition from negligible to substantial increase in the inter-packet spacing and produce the input rate at that transition point as the available bandwidth estimate of a network path. It is important to note that iterative methods do not estimate the capacity or the cross-traffic rate at the tight link, nor do they use this information in their measurement process.

The second category is classified as *direct probing* and includes Delphi [23], IGI [9], and Spruce [26]. These methods essentially sample the cross-traffic at the tight link using packet-trains with a fixed input rate that is higher than the available bandwidth and estimate the cross-traffic rate λ_t:

$$\lambda_t = \frac{C_t E[y] - s}{x}, \tag{1}$$

where C_t is the tight-link capacity, $E[y]$ is the average output dispersion, x is the input dispersion, and s is the probe-packet size. The available bandwidth is then computed as $C_t - \lambda_t$. Clearly, direct probing methods must use the tight-link capacity C_t in their measurement algorithms. Current techniques assume that the tight link is the same as the narrow link and that it can be measured using an existing capacity estimation tool such as Pathrate.

Different from the above studies, Pathneck [8] is a recent router-based sampling technique that focuses on locating the tight link of the path instead of computing accurate available bandwidth. Pathneck introduces a probing technique called *recursive packet train*, in which the sender places a number of small packets on both sides of each probe-train and forces them to be dropped at certain routers along the path. The server then examines the time gap between each pair of TTL-expired messages generated by the same router and infers the location of the tight link.

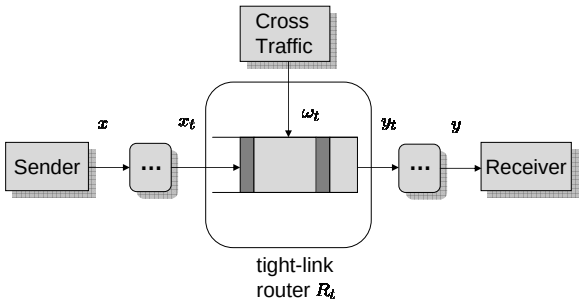


Fig. 1. Modeling an end-to-end path.

C. Single Link Models and Common Notation

Many studies in the literature discussed similar bandwidth estimation models (under different assumptions on cross-traffic) using a single-hop path. We summarize below the most important results that serve as a foundation for Envelope.

Fig. 1 schematically illustrates an end-to-end Internet path, where routers are present before and after the tight link[3] R_t. The measurement sender injects probe packet-trains with inter-packet spacing x and the receiver samples inter-arrival dispersion y, which is a random variable. Due to expansion/compression in pre-tight links, inter-packet spacing x_t of the probe packets entering router R_t is different from the initial spacing x. Then, the spacing x_t is altered at the router R_t to be y_t by random noise ω_t introduced by cross-traffic. Along the remaining routers, y_t is further "corrupted" to become y, which is what the receiver samples.

Recall that in the single-node analysis, it is assumed that cross-traffic in pre- and post-tight links does not change inter-packet spacings of the probe packets. Hence, the inter-arrival spacing x_t at the router R_t is trivially equal to the initial spacing x; and similarly, we have $y = y_t$. Based on this simple end-to-end path model, we next briefly review several previous single-hop results.

Melander *et al*. [21] and Dovrolis *et al*. [4] rely on a *constant-rate fluid* model of cross-traffic and obtain the following result

$$y = \max\left(x, \frac{s + x\lambda_t}{C_t}\right) = \begin{cases} x & x \geq \dfrac{s}{C_t - \lambda_t} \\ \dfrac{s + x\lambda_t}{C_t} & x \leq \dfrac{s}{C_t - \lambda_t} \end{cases}, \quad (2)$$

where λ_t is a (fixed) cross-traffic arrival rate at the tight link.

Note that in real networks such as the Internet, cross-traffic is bursty with a time-varying arrival rate. Considering the time-varying nature of cross-traffic, Kang *et al.* [14] derive the mean output dispersion under arbitrary cross-traffic when the input spacing $x \leq s/C_t$:

$$E[y] = \frac{s + x\lambda_t}{C_t}, \quad (3)$$

where λ_t is the time-average of a cross-traffic arrival rate process[4] $r_\tau(t)$ at the tight link:

$$\lambda_t = \lim_{t\to\infty} \frac{1}{t} \int_0^t r_\tau(u)du. \quad (4)$$

This result confirms the fluid model (2) as a valid first-order approximation of the real situation when the input spacing x is small. Another important result in [14] shows that the variance of y decays to 0 as the packet-train length N (i.e., the number of packets in each train) increases.

To extract both capacity C_t and available bandwidth A_t from $E[y]$, the paper [14] defines W_n^a and W_n^b to be the average output dispersion of two sets of measurements $\{y_i^a\}$ and $\{y_i^b\}$ (where the index i represents the packet-train sequence number) with different initial spacings x_a and x_b:

$$W_n^a = \frac{1}{n}\sum_{i=1}^{n} y_i^a, \;\; W_n^b = \frac{1}{n}\sum_{i=1}^{n} y_i^b \quad (5)$$

and derives asymptotically accurate estimators for the tight-link capacity and available bandwidth of a single-hop path:

$$\lim_{n\to\infty} \frac{s(x_a - x_b)}{x_a W_n^b - x_b W_n^a} = C_t, \quad (6)$$

$$\lim_{n\to\infty} s\left(\frac{x_a - x_b - W_n^a + W_n^b}{x_a W_n^b - x_b W_n^a}\right) = C_t - \lambda_t = A_t. \quad (7)$$

More recently, Liu *et al.* [18], [19] extend the single-hop analysis to accommodate arbitrary input spacings. Their results show that when $x > s/C_t$, the mean output dispersion $E[y]$ is larger than the fluid prediction in (2). However, as packet-train length N increases, the output dispersion y converges to (2) in the mean-square sense, which means that not only $E[y]$ converges to (2) but also $Var[y]$ decays to 0. This result allows us to apply a recursive extension of (6) and (7) to multi-hop paths.

III. Envelope

In this section, we first extend the single-hop estimators in [14] to multi-hop paths. We then discuss the phase-based probing technique used in Envelope.

A. Recursive Model

Recursive extension of the previous results to the multi-hop case can be performed by treating inter-packet spacing x_k of probe traffic arriving at router R_k as the inter-departure delays y_{k-1} of the previous router R_{k-1}:

$$x_k = y_{k-1}. \quad (8)$$

The recursive relationship between the average output dispersions $E[y_k]$ and $E[y_{k-1}]$, when a packet-train is sufficiently large, can be expressed as

$$E[y_k] = \begin{cases} E[y_{k-1}] & E[y_{k-1}] \geq \dfrac{s}{A_k} \\ \dfrac{s + \lambda_k E[y_{k-1}]}{C_k} & E[y_{k-1}] \leq \dfrac{s}{A_k} \end{cases}. \quad (9)$$

[3]In the paper, "router" and "link" are used interchangeably.

[4]The term $r_\tau(t)$ represents the cross-traffic arrival rate during the time interval $[t, t+\tau]$, $\tau > 0$.

The intuition behind (9) is as follows. When the packet-train is large, the single-hop fluid prediction (2) becomes a valid first-order characterization of the stochastic response curve[5] given that the input spacing is fixed. Furthermore, large packet-trains reduce the variance of y_{k-1}, so that most of the input spacing samples at R_k are clustered in the neighborhood of $E[y_{k-1}]$, which allows the single-hop result derived using fixed input spacings to hold for the case that the input spacings are random.

It is easy to see from (9) that a necessary condition for the bandwidth characteristics of link R_k to be measurable is $E[y_{k-1}] < s/A_k$. Otherwise, R_k preserves average incoming inter-packet spacings and thus $E[y_k]$ contains no information about the capacity or available bandwidth of the link. If this necessary condition can be satisfied by adjusting the initial input spacing x at the sender, we say that R_k is *measurable*. It is possible sometimes that the condition $E[y_{k-1}] < s/A_k$ cannot be satisfied no matter how small x is. In such case, R_k is *unmeasurable*. The following fact re-states the above discussion.

Fact 1: The tight-link router R_t is always measurable as long as the initial spacing $x < s/A_t$.

We can also see from (9) that $E[y_k]$ at any router R_k is monotonically non-decreasing. Hence, to make as many links fall into the measurable category as possible, we should use the initial spacing x as small as it can be (we discuss more about this issue in the next subsection).

When a particular router R_k is measurable, we must obtain both $E[y_{k-1}]$ and $E[y_k]$ so that we can apply the second part in (9) to compute λ_k and C_k. The next fact states the condition necessary to preserve $E[y_k]$ in the path suffix from router R_{k+1} to the receiver.

Fact 2: The mean inter-departure delay $E[y_k]$ of router R_k is preserved along the path suffix from link R_{k+1} up to the receiver, i.e., $E[y] = E[y_k]$, as long as the condition $E[y_k] > s/A_i$ holds for all $i > k$.

Fact 2 implies that for the receiver to obtain $E[y_k]$, the average output rate $s/E[y_k]$ of the packet-train at router R_k must be smaller than the available bandwidth of all downstream links along the path.

Summarizing the above results, we obtain that in order to measure link R_k, the spacing between the probe packets must be small when arriving at router R_k; however, in order to preserve the mean of the sampled signal along the path suffix, the departure spacing must be large. Since these two requirements contradict each other, we next develop a methodology called Envelope that preserves $E[y_k]$ based on Fact 2, but at the same time satisfies the condition in Fact 1.

B. Envelope Packet-Train

To obtain departure spacing from router R_k, we use a probe-train of N packets surrounded by two *envelope* packets E_1 and E_2 as demonstrated in Fig. 2. As the figure shows, all probe packets $P_1, \ldots, P_N$ are dropped at router R_{k+1} (using TTL limiting) and the surviving envelope packets carry spacing that is $N+1$ times larger than y_k.

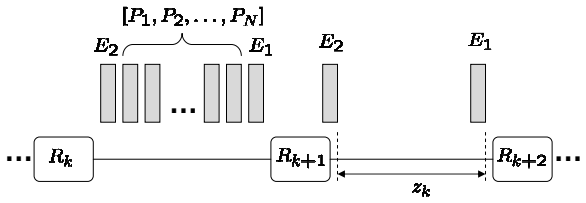


Fig. 2. A probe-train $[P_1, \ldots, P_N]$ of N packets enveloped by two packets E_1 and E_2 at router R_k.

Before continuing further, we should address several aspects of this approach. First, as in other packet-train methods (e.g., [4], [8], [21]), there is a chance that Envelope may congest the network by sending trains of $N+2$ back-to-back packets. While this drawback appears unavoidable in all such methods, it is important to remember that the delay *between* the trains can be arbitrarily large and that Envelope can maintain any desired average probing rate. Second, Envelope does not require any special "cooperation" from the routers (such as ICMP TTL-expired replies) since *all* routers must discard packets with $TTL = 0$ to prevent infinite looping of packets.

To satisfy the condition in Fact 2, the dispersion between the two envelope packets must be sufficiently large, which can be accomplished in practice by properly selecting a large N[6]. After the probe-train is dropped at router R_{k+1}, the receiver samples spacing z_k between envelope packets E_1 and E_2. Since we assume that z_k is large enough to be preserved along the path suffix, we directly obtain:

$$E[z_k] = (N+1)E[y_k]. \tag{10}$$

Define $W_{n,k}$ to be a normalized average of n samples of $z_{i,k}$ (where i is the packet-train sequence number as before) with respect to a given router R_k:

$$W_{n,k} = \frac{1}{n}\sum_{i=1}^{n}\frac{z_{i,k}}{N+1}. \tag{11}$$

Then, similar to the single-hop case in [14], the bandwidth of R_k can be inferred by using two sets of measurements $\{z^a_{i,k}\}$ and $\{z^b_{i,k}\}$ with two different initial inter-packet spacings $x = a$ and $x = b$ and computing the corresponding metrics $W^a_{n,k}$ and $W^b_{n,k}$ at the receiver:

$$W^a_{n,k} = \frac{1}{n}\sum_{i=1}^{n}\frac{z^a_{i,k}}{N+1}, \quad W^b_{n,k} = \frac{1}{n}\sum_{i=1}^{n}\frac{z^b_{i,k}}{N+1}. \tag{12}$$

Similar to (6) and (7), the capacity of link R_k can be estimated as:

$$\lim_{n\to\infty}\frac{s(W^a_{n,k-1} - W^b_{n,k-1})}{W^a_{n,k-1}W^b_{n,k} - W^b_{n,k-1}W^a_{n,k}} = C_k, \tag{13}$$

[5]The stochastic response curve represents a relationship between the statistical mean of packet-train output dispersions and the input dispersion [19].

[6]Recall that a large N is also needed to ensure the validity of (9).

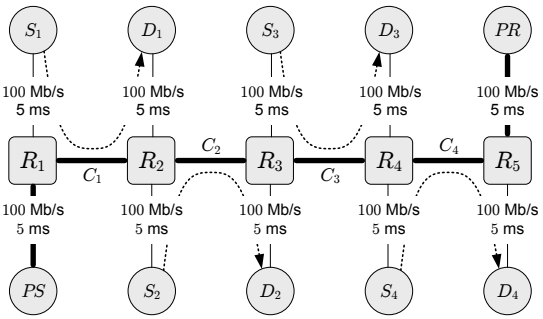


Fig. 3. Simulation topology.

and the available bandwidth $A_k = C_k - \lambda_k$ is:

$$\lim_{n\to\infty} s\left(\frac{W^a_{n,k-1} - W^b_{n,k-1} - W^a_{n,k} + W^b_{n,k}}{W^a_{n,k-1}W^b_{n,k} - W^b_{n,k-1}W^a_{n,k}}\right) = A_k. \quad (14)$$

Note that in both (13) and (14), the two metrics $W^a_{n,k-1}$ and $W^b_{n,k-1}$ have been obtained in the previous phase when link R_{k-1} was measured.

IV. Performance of Envelope

To evaluate the performance of Envelope, we present simulation results of Envelope including its estimation accuracy and asymptotic behavior and then compare its estimation accuracy with that of existing methods. We start by describing the simulation setup.

A. Simulation Setup

For simulations, we use the `ns2` network simulator with the topology shown in Fig. 3, in which source PS sends prob data to the destination PR through five router nodes $R_1, \ldots, R_5$. Cross-traffic is injected into each router R_i at an average rate λ_i through nodes S_i (where $i = 1, 2, 3, 4$). The speed of all access links is 100 Mb/s (delay 5 ms) and the remaining links L_i ($i = 1, 2, 3, 4$) between routers R_i and R_{i+1} have capacities C_i and propagation delay 10 ms.

To examine the estimation accuracy of Envelope, we use five different network settings shown in Table I, which lists the capacity and available bandwidth of each link for different simulation scenarios. The darkly shaded values in each row represents the tight-link capacity and available bandwidth of the path for each case. We also lightly shade the narrow link capacity in cases when it is different from the tight link. Note that in cases I and III, the narrow link *coincides* with the tight link; in cases II and IV, the narrow link *follows* the tight link; while in case V, the narrow link *precedes* the tight link.

All simulations are partitioned into two categories: CBR and TCP cross-traffic. For the first scenario, 10 UDP sources are attached to each node S_i to generate CBR flows that are injected into each router R_i. Each CBR flow starts with a random initial delay and uses 500-byte packets[7]. For simulations with TCP cross-traffic, we attach 50 FTP sources to each of S_i and

[7]We remind that the aggregated cross-traffic that actually traverses each router is not CBR but an aggregate CBR flow with rather bursty characteristics.

TABLE I
Simulation Setup

	Different link bandwidths (Mb/s)							
	C_1	A_1	C_2	A_2	C_3	A_3	C_4	A_4
Case-I	5	1	100	50	100	40	1.5	0.3
Case-II	2	0.4	1.5	0.25	0.8	0.4	1.5	0.35
Case-III	1.5	0.3	100	50	100	40	5	1
Case-IV	20	4	15	2.5	8	4	15	3.5
Case-V	2	0.4	0.8	0.4	1.5	0.25	2	0.4

keep the utilization of each router R_i according to the values shown in Table I. To maintain a fixed average utilization at each link in the TCP scenario, we placed an additional router (not shown in the figure) between each node S_i and router R_i to limit the aggregate sending rate of the TCP flows to the capacity of the additional router. The utilization of R_i is controlled by properly setting the capacity of the auxiliary router. The TCP cross-traffic consists of a mixture of flows with packet sizes 640, 840, 1040, 1240, and 1440 bytes. In both TCP and UDP scenarios, link utilization along the path of probe packets varies between 50% and 83%.

The probe-traffic sender PS sends packet-trains of length $N = 80$ with 40-byte packets at an average rate of 50 kb/s. The inter-packet spacing in alternate packet-trains is initialized to two different values x_a and x_b as described in the previous section.

Note that every simulation has four phases, one for each link in Fig. 3. In each phase ϕ_k ($k = 1, 2, 3, 4$), the packet-trains enclosed by envelope-packets are dropped at router R_{k+1}, while the two envelope packets are forwarded up to the receiver. In each phase, the receiver PR computes the average spacing $W^a_{n,k}$ and $W^b_{n,k}$ after receiving $n = 100$ packet-trains for each of x_a and x_b. The two average spacings are then applied to (13) and (14) to produce estimates of capacity and available bandwidth of the router under investigation.

B. Estimation Accuracy of Envelope

We next investigate the properties of Envelope, its estimation accuracy, and convergence behavior. We first define the following relative error metrics:

$$e_{C_i} = \frac{|C_i - \tilde{C}_i|}{C_i}, \quad e_{A_i} = \frac{|A_i - \tilde{A}_i|}{A_i}, \quad (15)$$

where e_{C_i} and e_{A_i} are the relative estimation errors of C_i and A_i, respectively, C_i is the true capacity of a link L_i, $\tilde{C}_i$ is its estimate, A_i is the true available bandwidth of L_i, and $\tilde{A}_i$ is its estimate.

Simulation results of Envelope are summarized in Tables II and III, which show relative estimation errors e_{C_i} and e_{A_i} under CBR and TCP cross-traffic, respectively. In the tables, empty cells represent links that are not measurable by Envelope. As Table II shows, under CBR cross-traffic, Envelope correctly identifies the tight link (shaded in the table) and computes its capacity with over 95% accuracy as well as the available bandwidth with 95%−99% accuracy. Under TCP cross-traffic, Envelope also properly locates the tight link and

TABLE II

PERFORMANCE OF ENVELOPE (CBR CROSS-TRAFFIC)

	Relative estimation error				
	Case-I	Case-II	Case-III	Case-IV	Case-V
e_{C_1}	0.94%	2.39%	0.17%	0.15%	10.76%
e_{A_1}	7.75%	1.57%	3.74%	6.99%	4.20%
e_{C_2}	—	0.35%	—	2.36%	2.47%
e_{A_2}	—	2.09%	—	5.62%	8.71%
e_{C_3}	—	3.76%	—	0.65%	4.13%
e_{A_3}	—	7.07%	—	2.04%	5.71%
e_{C_4}	1.56%	0.60%	—	12.11%	21.19%
e_{A_4}	2.38%	3.05%	—	9.86%	17.59%

TABLE III

PERFORMANCE OF ENVELOPE (TCP CROSS-TRAFFIC)

	Relative estimation error				
	Case-I	Case-II	Case-III	Case-IV	Case-V
e_{C_1}	0.21%	0.40%	0.46%	8.55%	4.22%
e_{A_1}	12.07%	0.27%	0.98%	21.23%	16.60%
e_{C_2}	—	3.62%	—	0.26%	5.90%
e_{A_2}	—	4.22%	—	3.29%	10.06%
e_{C_3}	—	10.79%	—	9.41%	10.06%
e_{A_3}	—	15.44%	—	23.30%	5.82%
e_{C_4}	0.24%	10.04%	—	—	—
e_{A_4}	3.30%	11.53%	—	—	—

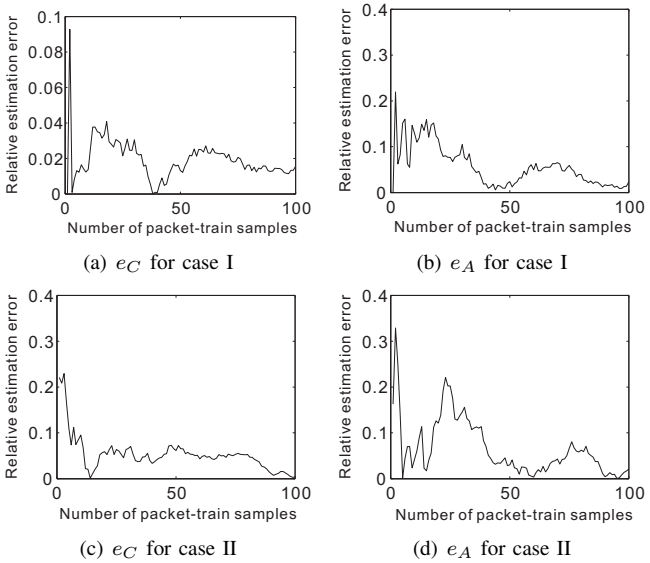
produces its bandwidth estimates $\tilde{C}$ and $\tilde{A}$ with $90\% - 99\%$ accuracy as shown in Table III.

In what follows below, we examine the asymptotic behavior of Envelope and show the evolution of its estimates. Figs. 4 and 5 plot the evolution of Envelope's relative estimation errors e_C and e_A (of the tight link) under CBR cross-traffic. As the figures show, the tight link capacity estimate $\tilde{C}$ converges to within 5% of its true value for each case after 100 packet-train samples. Note that e_A also becomes less than 5% after convergence takes place. For TCP cross-traffic, we observe a similar asymptotic behavior for e_C and e_A in all studied cases (see Figs. 6 and 7).

C. Available Bandwidth Comparison

In this subsection, we compare Envelope with several existing methods with respect to estimation accuracy using the setup shown in Table I.

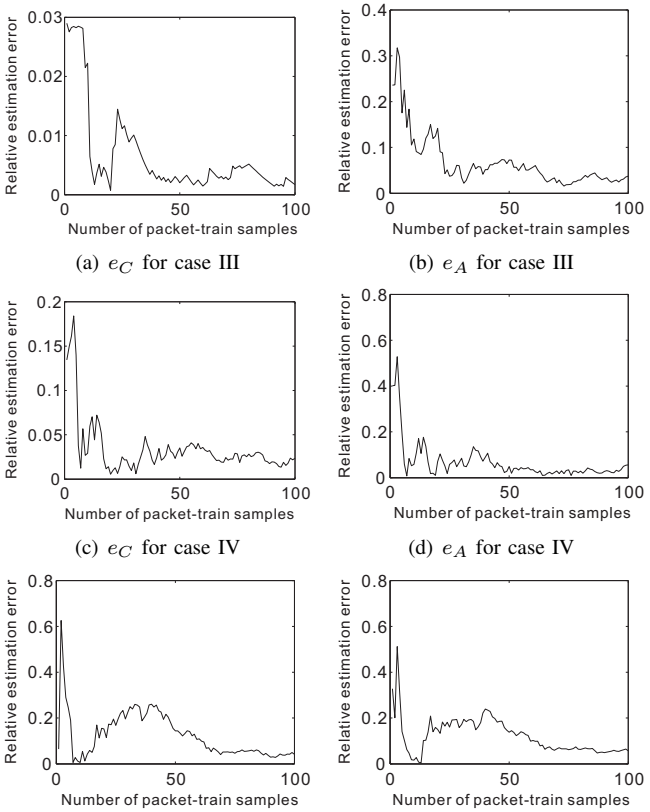
We first compare Envelope with several recent available bandwidth estimation methods Pathload [12], Spruce [26], and IGI [9]. We conduct all simulations over mildly to heavily loaded links with utilization varying between 50% and 83%.



(a) e_C for case I (b) e_A for case I

(c) e_C for case II (d) e_A for case II

Fig. 4. Evolution of relative estimation errors e_C and e_A for cases I and II under CBR cross-traffic.



(a) e_C for case III (b) e_A for case III

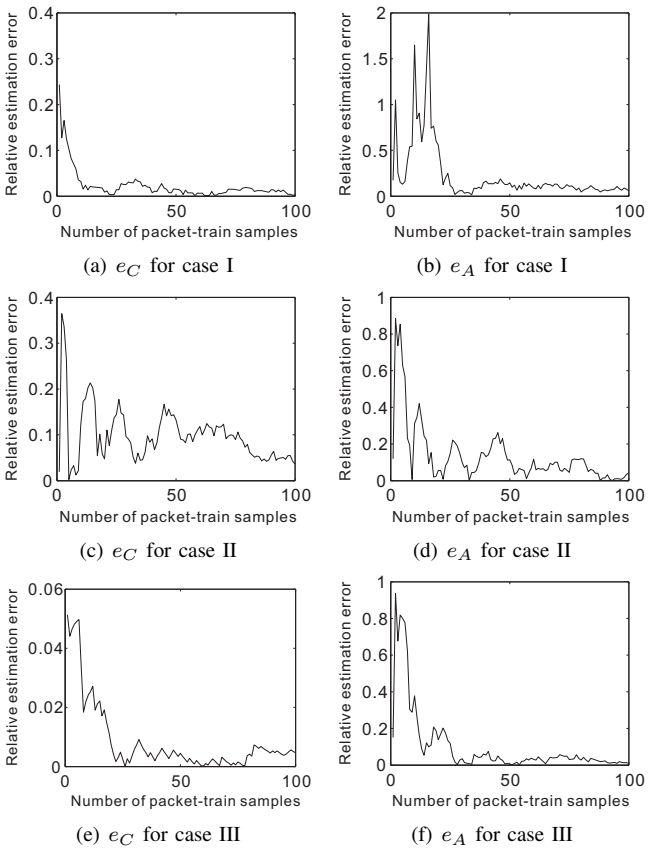
(c) e_C for case IV (d) e_A for case IV

(e) e_C for case V (f) e_A for case V

Fig. 5. Evolution of relative estimation errors e_C and e_A for cases III, IV, and V under CBR cross-traffic.

Note that with negligible cross-traffic interference at non-tight links, all methods produce accurate estimates of available bandwidth. However, when cross-traffic is non-negligible, the estimation accuracy is drastically different depending on the applied methods.

Tables IV and V show relative estimation errors e_A for the different cases under CBR and TCP cross-traffic, respectively. For Pathload, we use up to 9600 samples with a very fine-grained bandwidth resolution (4% of available bandwidth) and average the low and high values of the produced estimates. For Spruce, we use the last 100 samples to obtain the main estimate as suggested in [26]; and in the IGI case, we use the estimates available at the end of IGI's internal convergence



(a) e_C for case I (b) e_A for case I

(c) e_C for case II (d) e_A for case II

(e) e_C for case III (f) e_A for case III

Fig. 6. Evolution of relative estimation errors e_C and e_A for cases I, II, and III under TCP cross-traffic.

algorithm. Also note that we feed both Spruce and IGI the exact bottleneck capacity C, while Envelope and Pathload operate without this information.

As both tables show, Envelope significantly outperforms Spruce and IGI in all examined scenarios. Note that Pathload also produces much more accurate bandwidth estimates than Spruce and IGI and its results are comparable to those of Envelope.

D. Bottleneck Bandwidth Comparison

Since in all five paths simulated, the narrow link is measurable, we next compare Envelope with recent bottleneck bandwidth estimators CapProbe [15] and Pathrate [4], both of which provide very accurate capacity estimates in lightly congested paths. CapProbe [15] suggests to use 100 samples for estimation; however, we find that with 100 samples this algorithm does not converge, nor does it produce accurate estimates in our multi-link topology given cross-traffic conditions in Table I. Hence, we run simulations for 2000 seconds and use up to 10000 samples for better accuracy. In Pathrate, the internal algorithm executes for over 900 seconds and uses up to 11440 samples to get estimates of the bottleneck capacity of the end-to-end path.

Tables VI and VII illustrate relative capacity estimation errors e_C of the different methods under CBR and TCP cross-traffic, respectively. As the tables show, Envelope produces

TABLE IV

AVAILABLE BANDWIDTH ESTIMATION METHODS (CBR CROSS-TRAFFIC)

	Relative estimation error			
	Envelope	Pathload	Spruce	IGI
Case-I	2.38%	10.07%	33.99%	38.67%
Case-II	2.09%	23.69%	68.78%	64.25%
Case-III	3.74%	10.00%	13.91%	39.13%
Case-IV	5.62%	6.20%	64.69%	53.54%
Case-V	5.71%	12.45%	65.65%	44.40%

TABLE V

AVAILABLE BANDWIDTH ESTIMATION METHODS (TCP CROSS-TRAFFIC)

	Relative estimation error			
	Envelope	Pathload	Spruce	IGI
Case-I	3.30%	8.33%	34.83%	86.67%
Case-II	4.22%	12.09%	78.00%	109.20%
Case-III	0.98%	3.33%	7.65%	103.05%
Case-IV	3.29%	15.88%	78.15%	98.96%
Case-V	5.82%	12.04%	70.85%	91.60%

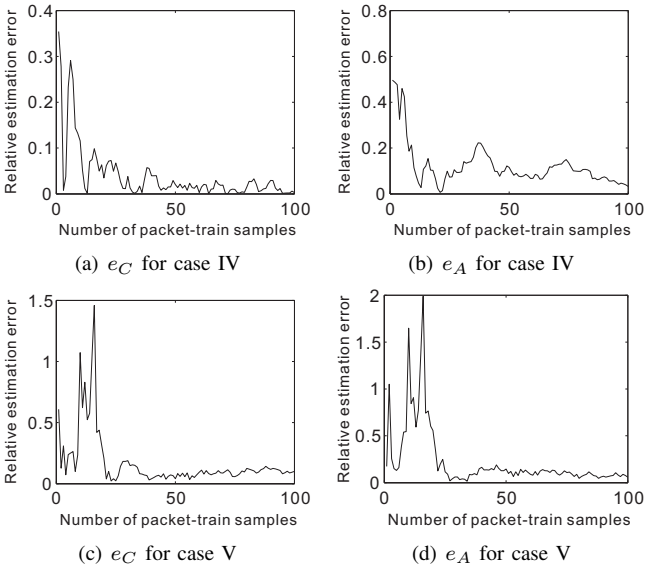
significantly better capacity estimates than CapProbe and Pathrate.

V. ANALYSIS OF EXISTING METHODS

In this section, we examine bandwidth sampling techniques used in several existing methods (Spruce, IGI, and CapProbe) and understand the reasons for their estimation inaccuracy.

A. Spruce and IGI

Note that even with the exact bottleneck capacity information, Spruce and IGI produce estimates with very high relative errors (see Tables IV and V). Recall that Spruce is based on the probe gap model (PGM) [26], which is derived under the assumption of a single bottleneck link that is both the narrow



(a) e_C for case IV (b) e_A for case IV

(c) e_C for case V (d) e_A for case V

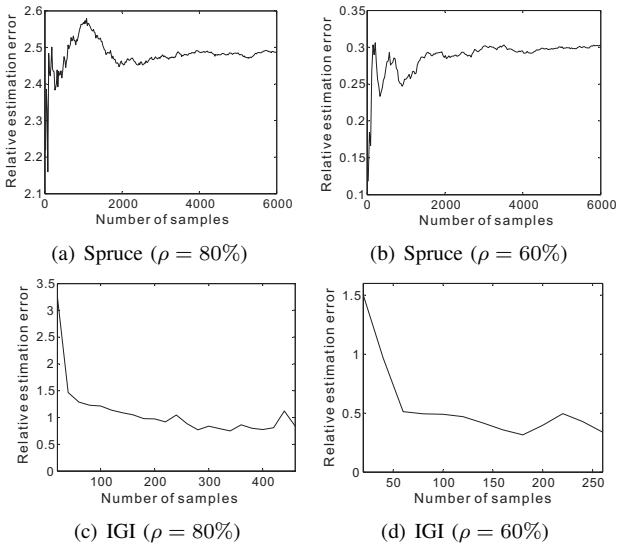
Fig. 7. Evolution of relative estimation errors e_C and e_A for cases IV and V under TCP cross-traffic.

TABLE VI
CAPACITY ESTIMATION METHODS (CBR CROSS-TRAFFIC)

	Relative estimation error		
	Envelope	CapProbe	Pathrate
Case-I	1.56%	40.00%	40.53%
Case-II	3.76%	43.75%	43.75%
Case-III	0.17%	38.46%	40.47%
Case-IV	0.65%	48.86%	43.75%
Case-V	2.74%	50.03%	42.25%

TABLE VII
CAPACITY ESTIMATION METHODS (TCP CROSS-TRAFFIC)

	Relative estimation error		
	Envelope	CapProbe	Pathrate
Case-I	0.24%	40.95%	40.93%
Case-II	10.79%	39.12%	32.50%
Case-III	0.46%	35.78%	48.10%
Case-IV	9.41%	50.60%	20.62%
Case-V	5.90%	51.62%	45.62%



(a) Spruce ($\rho = 80\%$) (b) Spruce ($\rho = 60\%$)

(c) IGI ($\rho = 80\%$) (d) IGI ($\rho = 60\%$)

Fig. 8. Evolution of relative available bandwidth estimation error e_A of Spruce and IGI for different link utilization ρ.

and the tight link along the path. By measuring packet spacing at the receiver, Spruce collects individual samples A_i [26]:

$$A_i = C\left(1 - \frac{y_i - x}{x}\right), \tag{16}$$

where x is the initial inter-packet spacing at the sender and y_i is the i-th measured packet spacing at the receiver. The algorithm averages samples A_i to obtain a running estimate of the available bandwidth $A_n = \sum_{i=1}^{n} A_i/n$.

Note that this method does not take into account interference of cross-traffic with the probe-gap at the routers other than the bottleneck router over the entire path. Hence, with congested pre- and post-bottleneck links, Spruce's estimation accuracy degrades significantly. For example, in cases II, IV, and V, the estimation error is over $60 - 70\%$ as shown in Tables IV and V. To illustrate the estimation accuracy of Spruce for different link utilization ρ, we extract the evolution of Spruce's estimate A_n in case II with TCP cross-traffic. For this demonstration, we set the utilization of all links in the path to ρ and plot the relative estimation errors e_A for $\rho = 80\%$ and $\rho = 60\%$ in Figs. 8(a) and 8(b), respectively. As the figures show, the convergence error of e_A is reduced from 245% to 30% when ρ is lowered from 80% to 60%. This explains the high estimation errors exhibited in Tables IV and V and confirms Spruce's limitations in heavily-loaded multi-link paths.

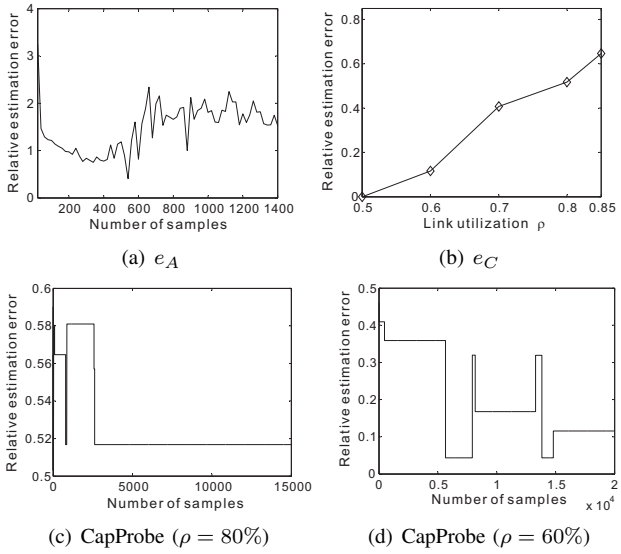
IGI [9] is also based on a probing gap model. IGI sends a sequence of packet-trains with increasing inter-packet spacing in each packet-train until it reaches the *turning point* (where the initial inter-packet spacing x at the sender equals the inter-packet spacing y at the receiver). This method assumes that at the turning point, the noise introduced by cross-traffic becomes zero mean and the probing rate is equal to the available bandwidth of the path (which is not true [19]). Furthermore, the analysis in [9] with respect to a single congested node does not consider the interference of pre- and post-bottleneck cross-traffic, which randomly changes the probe-gap at the receiver. As a result, IGI's estimate can converge to a value that is significantly different from the true path available bandwidth.

To emphasize this fact, we simulate case II with TCP cross-traffic for two different utilizations $\rho = 80\%$ and $\rho = 60\%$. We plot in Figs. 8(c) and 8(d) the evolution of relative estimation errors e_A until IGI's internal algorithm terminates at $x/y = 0.9$. As the figure shows, IGI produces the available bandwidth estimate with 80% error in the heavily congested case ($\rho = 80\%$), while the same error is only 30% in the case of $\rho = 60\%$.

Moreover, IGI's estimation accuracy gets worse if the initial packet spacing x becomes close to the turning point. For illustration, we plot the evolution of IGI's available bandwidth estimate in Fig. 9(a) until it terminates its internal algorithm at $x/y = 0.999$. In the figure, IGI produces the best estimate some time before it reaches the turning point and thereafter the accuracy becomes worse as the initial spacing becomes closer to the turning point. Hence, unlike some of the other approaches studied in this work, using more samples in IGI does not necessarily lead to better estimation accuracy (for this reason, we let the IGI algorithm terminate at $x/y = 0.9$ to produce its best available bandwidth estimates discussed in Section IV-C).

B. CapProbe

Recall that CapProbe is based on the assumption that if packets in a probe pair have arrived at the receiver with the smallest combined one-way delay, then the packets have not been queued at any intermediate routers in the path and thus the inter-packet delay of the probe pair reflects the transmission delay of the bottleneck link. Based on this assumption, CapProbe uses 100 samples and the minimum delay condition to obtain the packet-pair that contains information about C. However, CapProbe's minimum filtering is sensitive to random



(a) e_A (b) e_C

(c) CapProbe ($\rho = 80\%$) (d) CapProbe ($\rho = 60\%$)

Fig. 9. (a) Evolution of relative estimation error e_A of IGI for $\rho = 80\%$. (b) Relative estimation error e_C of CapProbe for different utilization ρ of the links in case II. (c)-(d) Evolution of relative capacity estimation error e_C of CapProbe for two different values of ρ.

queuing delays in front of the first packet of the pair and thus it is possible that the estimated capacity converges to a value that is very different from the true value of C. Furthermore, the convergence is rather random so that it is difficult to decide the number of samples to be used in the estimation algorithm.

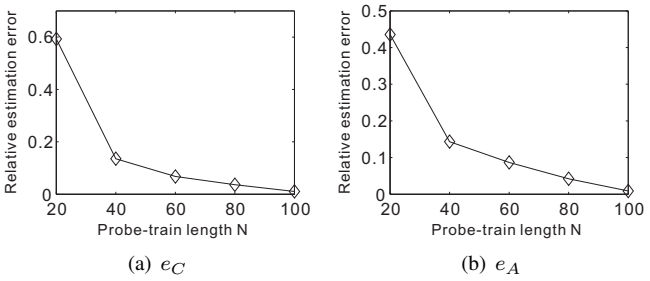
To illustrate the queuing effects discussed above, we simulate case II with TCP cross-traffic for varying link utilization ρ between 50% and 85%. Fig. 9(b) shows CapProbe's relative capacity estimation errors e_C for different ρ. Observe that as ρ of the path increases from 50% to 85%, the relative estimation error e_C jumps from 0% to 64%. This indicates that CapProbe performs very well in a lightly utilized path; however, in a heavily-congested path, its capacity estimation accuracy significantly degrades. Furthermore, CapProbe's estimation accuracy fluctuates substantially depending on the number of samples used as illustrated in Figs. 9(c) and 9(d). For example, in a path with $\rho = 60\%$, $e_C = 41\%$ with 100 samples, 4% with 6000 samples, 19% with 10000 samples, and 11% with 20000 samples. This indicates that if the measuring process stops at a random plateau where CapProbe has seemingly converged, its estimation accuracy will be random as shown in Fig. 9(d).

VI. Probing Parameters in Envelope

A. Initial Input Spacing

We now discuss how Envelope chooses the two initial input spacings x_a and x_b. Recall from Fact 1 that as long as the two spacings are less than s/A_t and differ from each other, the tight link can be measured. Note that Envelope can measure many non-tight links if the conditions in Facts 1 and 2 hold. Hence, it tries to choose initial spacing values that is much smaller than s/A_t. To achieve this, Envelope utilizes the known access link capacity C_0 and sets x_a as small as possible: $x_a = s/C_0$. Then, it probes for the Asymptotic Dispersion Rate (ADR) of the path to determine x_b. Envelope obtains ADR by sending a packet-train with spacing x_a and computes $ADR = s/E[y]$ at the receiver, which is similar to Pathrate. Based on ADR, Envelope determines the second spacing $x_b = s/(\alpha C_0 + (1-\alpha)ADR)$, where $0 < \alpha < 1$ is tunable.

It is proved in [4] that $A_t < ADR < C_0$, which confirms that our initial spacing settings satisfy the condition in Fact 1 discussed in Section III.



(a) e_C (b) e_A

Fig. 10. Relative estimation errors e_C and e_A of Envelope for varying probe-train length N in case II under TCP cross-traffic.

B. Probe-Train Length

We study the impact of probe-train length N on the estimation accuracy of Envelope. Recall that the main idea behind Envelope is to preserve the inter-departure spacing $y_{i,k}$ of probe packets in the path suffix of a congested router R_k using envelope packets. As suggested in Section III-B, maintaining a large inter-envelope packet spacing $z_{i,k}$ preserves its mean $E[z_{i,k}]$ in the path suffix since cross-traffic noise introduced into $z_{i,k}$ becomes zero-mean.

We conduct simulations using case II under TCP cross-traffic for different probe-train length N and plot the relative estimation errors e_C and e_A of the tight link in Fig. 10. As the figure shows, the estimation accuracy of Envelope is improved as probe-train length N increases. For example, Envelope produces the bottleneck capacity estimate with error 60% for $N = 20$; however, its error is reduced to 13% for $N = 40$ and 6% for $N = 60$ (see Fig. 10(a)). Similarly, with small train length $N = 20$, the accuracy of available bandwidth estimation is rather low (43% error), but when the train length increases to $N = 60$, Envelope estimates the available bandwidth with just 8% error as shown in Fig. 10(b).

C. Amount of Probe Data

We briefly discuss the amount of probe data used in bandwidth sampling for different methods. For existing methods, we used the same packet size and number of trains or packet pairs recommended in the original paper. Table VIII shows the number of packet samples and corresponding data used to get bandwidth estimates for cases I and III. As the table shows, Spruce, IGI, and CapProbe do not use many samples while Pathload, Pathrate, and Envelope requires significantly more probe packets for their measurement. Note that Envelope requires even more samples than Pathrate to examine individual links. However, unlike other methods, Envelope can use very small packets without sacrificing estimation accuracy. For

TABLE VIII
BANDWIDTH SAMPLING OVERHEAD FOR CASES I AND III

Method	Number of samples	Packet size (bytes)	Total probe data (MB)
Envelope	64,000	40	2.56
Pathrate	11,440	1,500	17
CapProbe	600	1,500	0.9
Pathload	4,200	192	0.81
Spruce	533	1,500	0.8
IGI	900	800	0.72

instance, Pathrate used 11440 samples in our study, which accounts to 17 MB of data, while Envelope used 2.56 MB of probe packets. Recall that the number of samples required for Envelope is proportional to the number of hops in the path. For a 10-hop Internet path, Envelope needs a maximum of 6.4 MB of probe data to examine all individual links. However, since Envelope can terminate the measurement right after the tight link is measured, the required number of probe packets will be often smaller than the maximum.

VII. CONCLUSION

This paper proposed a recursive extension of the bandwidth estimators in [14] to network paths with multiple routers and presented a new tight-link bandwidth estimation technique, called Envelope, for end-to-end network paths. Through simulations, we showed that Envelope is asymptotically accurate (to the extent possible to observe using finite-sampling) in estimating both types of bandwidth. We also presented extensive simulation results of existing methods Pathload, Spruce, IGI, Pathrate, and CapProbe and their comparison to Envelope. These results suggested that in multi-link paths with significant cross-traffic interference at non-tight links, most of existing methods cannot converge to the correct values of C or A, even if the sampling process is sufficiently long.

We further demonstrated that in many network settings, Envelope can also estimate bandwidth for some non-tight links in the path and locate the position of the tight link. Our Future work involves implementation of Envelope in the Internet and further reduction of traffic required to obtain C and A.

REFERENCES

[1] B. Ahlgren, M. Björkman, and B. Melander, "Network Probing Using Packet Trains," *Swedish Institute Technical Report*, 1999.

[2] M. Allman and V. Paxson, "On Estimating End-to-End Network Parameters," *ACM SIGCOMM*, 1999.

[3] R. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet Switched Networks," *International Journal on Performance Evaluation*, vol. 27 & 28, pp.297-318, 1996.

[4] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-Dispersion Techniques and a Capacity-Estimation Methodology," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963-977, 2004.

[5] A. Downey, "Using PATHCHAR to Estimate Internet Link Characteristics," *ACM SIGCOMM*, 1999.

[6] K. Harfoush, A. Bestavros, and J. Byers, "Measuring Bottleneck Bandwidth of Targeted Path Segments," *IEEE INFOCOM*, 2003.

[7] G. He and J. Hou, "On exploiting long range dependence of network traffic in measuring cross traffic on an end-to-end basis," in *IEEE INFOCOM*, March 2003.

[8] N. Hu, L. Li, Z. Mao, and P. Steenkiste, "Locating Internet Bottlenecks: Algorithms, Measurements, and Implications," *ACM SIGCOMM*, 2004.

[9] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879-974, August 2003.

[10] V. Jacobson, "pathchar – A Tool to Infer Characteristics of Internet Paths," ftp://ftp.ee.lbl.gov/pathchar/.

[11] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM*, 1988.

[12] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth," *Passive and Active Measurements (PAM) Workshop*, 2002.

[13] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls in end-to-end available bandwidth estimation," in *ACM IMC*, October 2004.

[14] S. Kang, X. Liu, M. Dai, and D. Loguinov, "Packet-Pair Bandwidth Estimation: Stochastic Analysis of a Single Congested Node," *IEEE ICNP*, 2004.

[15] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: a Simple and Accurate Capacity Estimation Technique," *ACM SIGCOMM*, 2004.

[16] K. Lai and M. Baker, "Measuring Bandwidth," *IEEE INFOCOM*, 1999.

[17] K. Lai and M. Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," *ACM SIGCOMM*, 2000.

[18] X. Liu, K. Ravindran, B. Liu, and D. Loguinov, "Single-Hop Probing Asymptotics in Available Bandwidth Estimation: Sample-Path Analysis," *ACM IMC*, 2004.

[19] X. Liu, K. Ravindran, and D. Loguinov, "Multi-Hop Probing Asymptotics in Available Bandwidth Estimation: Stochastic Analysis," *ACM IMC*, 2005.

[20] B. Mah, "pchar: A Tool for Measuring Internet Path Characteristics," *http://www.employees.org/£\sim£bmah/Software/pchar/, year = 2001.*

[21] B. Melander, M. Björkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," *IEEE GLOBECOM*, 2000.

[22] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," *Ph.D. dissertation, Computer Science Department, University of California at Berkeley*, 1997.

[23] V. Ribeiro, M. Coates, R. Riedi, S. S. B. Hendricks, and R. Baraniuk, "Multifractal Cross-Traffic Estimation," *ITC Special Seminar on IP Traffic Measurement, Modeling, and Management*, September 2000.

[24] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," *Passive and Active Measurements (PAM) Workshop*, 2003.

[25] V. Ribeiro, R. Riedi, and R. Baraniuk, "Locating available bandwidth bottlenecks."

[26] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," *ACM IMC*, 2003.