Empirical Evaluation of Techniques for Measuring Available Bandwidth

Alok Shriram and Jasleen Kaur
Department of Computer Science
University of North Carolina at Chapel Hill
Email: {alok,jasleen}@cs.unc.edu

***Abstract*— The ability to measure end-to-end Available Bandwidth (AB) on a network path is useful in several domains, including overlay-routing infrastructure, network monitoring, and design of transport protocols. Several tools have, consequently, been proposed to estimate end-to-end AB. Unfortunately, existing evaluations of these tools are either not comprehensive or are biased by the current state of implementation technology. In this paper, we conduct a comprehensive empirical evaluation of algorithmic techniques for measuring AB. In order to eliminate implementation-related biases, we rely on a simulated network environment and develop a generic implementation framework for instantiating different tools. We implement our framework in NS-2 and reproduce traffic from real Internet links in order to evaluate tools under different conditions of (i) traffic load, (ii) sampling intensities, (iii) measurement timescales, (iv) number of bottleneck links, and (iv) location of bottleneck. We evaluate the tools for their accuracy, run-time, overhead, intrusiveness, and impact on responsive traffic. Our results contradict some of those in prior work that does not eliminate implementation biases.**

I. Introduction

The ability to measure the end-to-end available bandwidth (AB) of a path has applications in several domains, including (i) the design of a quick ramp-up phase for transport protocols like TCP [1], (ii) in selecting the best of several Internet paths between two communicating end-hosts [2], and (iii) in detecting highly-congested bottlenecks while monitoring a network [3]. The problem of measuring end-to-end AB has, consequently, received considerable attention in the recent literature and several AB estimation tools—henceforth, referred to as ABETs—have been designed and implemented for this [4], [5], [6], [7], [8], [9].

While the design of ABETs has received considerable focus, the evaluation of these has not been addressed adequately. In Section II, we illustrate that existing evaluations are either non-comprehensive and biased, or are affected by implementation issues. In this paper, we address this limitation by conducting an extensive experimental study of existing techniques for measuring AB.[1] In order to study these in a manner independent of implementation concerns, we rely on a simulated network environment and a common reference implementation of the different algorithmic techniques used to measure AB. This approach proves useful as some of our findings contradict those in past work that instead evaluates publicly-available *implementations* of ABETs.

[1]This research was supported in part by NSF CAREER grant CNS-0347814 and NSF RI grant EIA-0303590.

In the rest of this paper, we discuss related work in Section II and our experimental methodology in Section III. We present our experimental results in Sections IV–VI and our conclusions in Section VII.

II. State of the Art

Tool	Probe Stream	Inference Metric
Pathload [5]	Equi-Spaced Train	One-way Delay
PathChirp [6]	Exponential spacing	Dispersion
Spruce [7]	Packet-Pair	Dispersion
IGI [4]	Packet Train	Dispersion
Iperf	TCP-Stream	Throughput
Cprobe	Packet Train	Receiving Rate

TABLE I
Available Bandwidth Estimation Tools

A. Tool Design

Several tools (ABETs) have been proposed recently for actively probing for the end-to-end AB on a given network path [4], [5], [6], [7], [10], [11], [12]. These tools typically operate by injecting specially-designed streams of probe packets onto the path, observing the end-to-end delays experienced by the probe packets, and then estimating the end-to-end AB from the observations. Existing literature focuses primarily on two aspects of tool design:

- *Algorithmic techniques for inferring end-to-end AB:* ABETs differ most significantly in the design choices they make along two dimensions: (i) the structure of a probe stream—for instance, while Abing [12] and Spruce [7] rely on using a packet-pair as a probe stream, Pathload [5] and PathChirp [6] rely on sending a packet train (uniformly and exponentially-spaced, respectively) in each probe stream—and (ii) the inference logic used for estimating AB from the observed end-to-end delays. Table I summarizes these for popular ABETs.
  Given the myriad of tools available and the diversity in the algorithmic techniques they use, it is natural to ask: *which algorithmic technique performs the best?*
- *Implementation techniques for achieving high time-stamping accuracy:* In current high-speed networks, variations in end-to-end delays may have an order of magnitude in the sub-millisecond range. Since most ABETs rely on measuring delay variations, high resolution and accuracy in time-stamping probe packets is crucial for ensuring the accuracy of the inferred AB. Current PC platforms, however, are incapable of guaranteeing these

due to multi-tasking and the use of mechanisms such as interrupt coalescence [5]. Most tool implementors work around this limitation using two techniques [5], [6], [7]: (i) they rely on OS support for detecting and discarding probe streams that appear to not have been time-stamped accurately; and (ii) they collect observations from several probe streams before converging on a robust estimate of AB. While such techniques do not differ much across current ABETs, these do impact tool performance significantly [13], [14]. It is important to note that as technology improves [15], the need for and the impact of these techniques on ABET performance is likely to diminish. It is, therefore, natural to ask: *to what extent does current implementation technology limit tool performance? In particular, how well would tools perform if technology advances in the future?*

The questions raised above have been partly addressed in the literature, as discussed next.

B. Tool Evaluation

Several tool proponents compare the performance of their tools against that of others under controlled lab settings as well as in Internet-wide experiments [4], [6], [7]. Robeiro *et. al.* [6] compare the performance of PathChirp to that of Pathload [5] and TOPP [11] in an emulated lab setting. They find that PathChirp is more accurate than TOPP and less intrusive than Pathload. Hu *et. al.* [4] compare the performance of IGI/PTR to that of Pathload and Iperf on 13 Internet paths of capacity within 100 Mbps. They observe that while the readings of the three tools match on some paths, they fluctuate on other. Since the actual AB of these paths were not known, tool accuracy was not verified. Strauss *et. al.* [7] compare the performance of Spruce to that of Pathload and IGI. They use SNMP data collected at five minute intervals to evaluate the accuracy of these tools on two 100 Mbps paths. They also compare the sensitivity of the tools to changes in AB by performing several experiments on the RON testbed. They find that IGI is inaccurate at high loads, and Spruce is more accurate and less intrusive than Pathload.

Unfortunately, such studies are not comprehensive in either the tools or the settings evaluated. All of the studies described above evaluate only a small (and different) sub-set of ABETs, depending on which ones were popular when the corresponding tool was proposed. Furthermore, these evaluations include only simple network and traffic scenarios—for instance, none of the above evaluate tool performance against responsive cross-traffic, in high-speed networks, or when the tight and narrow links are different.[2] As a result of these practices, the results are often not comprehensive and get inadvertently biased toward highlighting the salient features of the proposed tool.

A recent work [16] addresses the above limitation by evaluating in a controlled high-speed lab setting, several publicly-available implementations of existing tools.[3] It treats each implementation as a black-box and observes its accuracy, timeliness, and overhead. While this study provides a comprehensive and unbiased evaluation of the implementations, it suffers from two key limitations. First, since it relies on a *black-box* approach, it does not provide insight into the performance of each tool as a function of its configurable parameters. Second, the observations are affected by the tool implementation efficiencies—such as dealing effectively with interrupt coalescence, context switches, and outliers—as well as the time-stamping precision of existing PC technology. It provides little insight into the relative performance of the algorithmic components of ABETs. For instance, [16] concludes that Pathload and PathChirp are the most accurate tools and packet-pair based tools such as Spruce and Abing do not perform well—it is not clear, however, if this result is fundamental to the packet-pair based inference logic, or is an artifact of the fact that these are less robust to time-stamping inaccuracies. In particular, as technology evolves and time-stamping resolutions improve, would such techniques continue to perform poorly? In this paper, we address the question—*in the absence of implementation-related limitations, how well do existing AB estimation techniques perform?*[4]

In [19], the authors analyze at large time-scales, the performance of several bandwidth estimators that can be represented mathematically. Unfortunately, their evaluation can not model several iterative estimators and is limited to low-bandwidth paths with a single bottleneck link.

Another significant limitation of *all* previous ABET evaluations is that these ignore the impact of two key probing-related quantities on the performance of ABETs. The first is the *measurement timescale* (MT), defined as the time-scale at which AB is observed. Practically, the MT used by an ABET corresponds to the duration of a single probe stream. The second is the *sampling intensity* (SI), which refers to the duration for which the AB is sampled per unit time. Practically, this is the product of the MT and the number of probe streams sent per unit time. It has been shown in [20] that both MT and SI impact the accuracy and variability of the AB sampled by any ABET. Unfortunately, none of the existing ABETs allow the choice of MT, and only some allow the choice of SI. Given the observations made in [20], it is important to *redesign the ABET interfaces to allow the choice of MT and SI, and study the impact of these on ABET performance.*

To summarize, existing ABET evaluations are either biased by limitations of current implementation technology and/or are

[2]The *tight* link of a path is one with the least amount of available bandwidth, while the *narrow* link is the one with the least transmission capacity [5]. The tight link of a path may not be the same as the narrow link if it carries significant amount of traffic load.

[3]Coccetti *et. al.* [17] evaluate early ABETs, including Iperf and Pathload, on a low speed (less than 4 Mbps) 4-5 hop topology with and without cross-traffic. They conclude that tool results strongly depend on configuration of the router queues and that a considerable amount of care would need to be taken while interpreting the results from any ABET, especially if QoS features were present in the network.

[4]The need for an implementation-independent evaluation of ABETs has also been highlighted in [18]—however, no experimental results have been made available to date.

not comprehensive in evaluating tools against diverse network and probing conditions.

C. Our Approach

We address the above-mentioned limitations in ABET evaluation using a two-pronged approach:

a) Evaluation independent of current implementation technology: We use a *simulation* environment to implement and evaluate the performance of prominent ABETs. By doing so, we ensure that the only variables that impact the performance of an ABET are the algorithmic design of its probe-stream and inference logic. Specifically, issues related to time-stamping accuracy, timer granularity, CPU load, and interrupt-processing are taken out of the equation—a simulator allows for perfect time-stamping and spacing of probe packets.

b) Evaluation against diverse probing and network conditions: We consider gigabit network paths and simulate diverse network conditions (specifically, RTT, traffic load, and number and location of bottleneck links). Additionally, we redesign the specification of existing ABETs to allow us to control and vary, whenever possible, probing parameters (specifically, MT and SI). We study the impact of all of the above parameters on the estimation accuracy and costs of each tool.

We incorporate realism in our evaluations by recreating the AB process from real Internet links (by replaying traffic traces captured from these links). We also evaluate the impact of ABETs on responsive cross-traffic by emulating at the application-level, the transmission behavior of end-applications that generate traffic in these link traces. In what follows, we describe our methodology and results in detail.

III. Experimental Framework

A. ABET implementation

We select several prominent ABETs—namely, Pathload [5], PathChirp [6], Spruce [7], IGI [4], Fast-IGI [4], and Cprobe [8]—that represent existing diversity in the algorithmic techniques used for inferring end-to-end AB. We implement each of these tools in the NS-2 [21] network simulation environment. We rely on published literature as well as publicly-available implementations (whenever available) to extract details of each tool.

Recall that one of our objectives is to study the impact of the probing parameters, MT and SI. Unfortunately, most tool designs do not allow for the choice of either of these two probing parameters. We address this limitation by redesigning the interfaces (and sometimes the specifications) of the ABETs, in order to allow us to select their MT and SI. We briefly describe this redesign for each ABET below—due to space constraints, we discuss only relevant aspects of each tool design; we refer the reader to the respective publications for details.

1) Incorporating MT: It is useful to observe that the MT of a tool is effectively given by the duration of individual probe streams—this is the duration for which the probe packets interact with the cross-traffic on the bottleneck link(s) and observe the AB process. In order to allow us to control the MT, we modify the tool interface designs as follows:

a) Cprobe/IGI/Fast-IGI: All three of these tools send several probe streams, each at a uniform rate, in order to estimate the AB—while Cprobe sends all streams at a high bit-rate, IGI/Fast-IGI iteratively change the bit-rate of each stream in order to converge on the AB. The number of packets sent within each probe stream is typically fixed. In order to incorporate MT, we make the number of packets (N) a *configurable* value that is set such that when the stream is sent at the desired bit-rate (R) and default packet size (P) (of 1500 Bytes), the stream duration is equal to MT: $N = \lceil R \times MT/P \rceil$.

b) Pathload: Like IGI, Pathload also iteratively sends several probe streams at different bit-rates. However, the Pathload AB inference logic requires that the number of packets sent in each probe stream be a perfect square. In order to incorporate MT, we first compute a rough estimate of N using the relation: $N = \lceil R \times MT/P \rceil$. If this is not a perfect square, we decrease P by the least amount required to ensure that it is.[5]

c) Spruce: Spruce relies on a packet-pair based AB inference technique, which sends two packets back-to-back in each probe stream. Unfortunately, this fixes the MT to a small value. However, due to the small probes and open-loop nature of Spruce, its run-time is fairly small; often smaller than the values of MT that may be of interest to us. We exploit this property to redesign Spruce's interface—given a desired MT, we continuously run Spruce several times and then compute the average of all AB estimations made within a duration of MT, in order to estimate the AB at that MT.

d) PathChirp: In PathChirp, each probe-stream—also referred to as a *chirp*—is an exponentially-spaced stream of N packets. The inter-packet spacing and N are determined using three parameters: the lower rate, L, the upper rate, U, and the spread-factor, S. Specifically, the spacing between packet i and $i+1$ is given by: $P/(L \times S^{i-1})$, and N is computed using the relation: $U = L \times S^N$. In order to incorporate MT, we first compute the length of a chirp as the following sum of a geometric series: $P/L \times (1 - \frac{1}{S^n})/(1 - \frac{1}{S})$. Given L and U, we then select the pair (S, N) such that the above chirp length is close to the desired MT.

2) Incorporating SI: The SI of a tool is effectively the fraction per unit time that is occupied by the probe streams it sends. If G is the gap between successive probe-streams, SI is given by: $\frac{MT}{MT+G}$. This relation can be used to control SI in open-loop tools such as Cprobe, Spruce,and PathChirp—specifically, given an SI, the gap is set to: $G = MT \times \frac{1-SI}{SI}$.

In *closed-loop* tools such as Pathload/IGI/Fast-IGI, however, the construction of a probe-stream is determined by the delays experienced by the previous probe-stream—these tools, therefore, can not send more than one probe-stream per RTT. Thus, the SI can not be set to a value higher than $MT/MT + RTT$,

[5]This convoluted way of controlling the MT in Pathload (and in PathChirp, as described later) highlights the limitation of existing ABET designs.

which is a fairly low value for typical Internet paths. Thus, for all practical purposes, SI can not be controlled in closed-loop tools. It is interesting to note, however, that the path RTT is likely to impact the feedback-loop and, hence, the performance (especially the run-time) of such tools.

B. Performance Metrics

We characterize the performance of each ABET using two types of metrics:

- *Accuracy-related:* Each run of an ABET should yield a good estimate of the end-to-end AB. In order to quantify the accuracy of an ABET estimate, we compute its *AB estimation error* as the difference between the estimated AB and the actual AB. The actual AB of a link is computed as the ratio of the number of bits that traverse the link during the tool run, to the tool run-time.
- *Cost-related:* We quantify the cost of using an ABET with several metrics. The *run-time* is defined as the time taken by a tool to return an estimate. The faster a tool runs, the better and valid its AB estimates are. Since we are relying on a simulation environment, this time is primarily governed by the number and sizes of probe-streams and the convergence logic used to estimate AB. For closed-loop tools, the run-time is also affected by the path RTT. The *probing overhead* is defined as the total amount of network probe traffic sent by the tool in order to arrive at a single estimate of AB. For large-scale deployment and use of ABETs, it is important that they use low amount of probe traffic. The *intrusiveness* is defined as the average bit-rate of a tool—this is given by the ratio of the overhead to the run-time. Since the run-times of ABETs can differ by orders of magnitude, it is important to compare the rate at which they inject probe traffic.

  In addition, we study the impact of probe traffic on the response time of ongoing TCP connections.

We conduct several types of experiments in order to study the above metrics—we describe these next.

IV. Validation

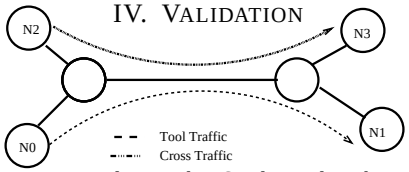


Fig. 1. Topology with a Single Bottleneck Link

The accuracy of most ABETs is typically established by their proponents by running them on links shared by cross-traffic with a *constant bit-rate* (CBR). We validate our NS-2 implementations of the selected ABETs by using the network topology depicted in Figure 1.[6] We run CBR cross-traffic between nodes N2 and N3, and instances of ABETs between nodes N0 and N1. We vary the cross-traffic load from 100 Mbps to 900 Mbps and for each load, we record the AB estimates from several back-to-back runs of each tool. It is important to note that all of our evaluations are conducted in high-speed gigabit networks—most ABET designs have not been evaluated in such a setting previously.

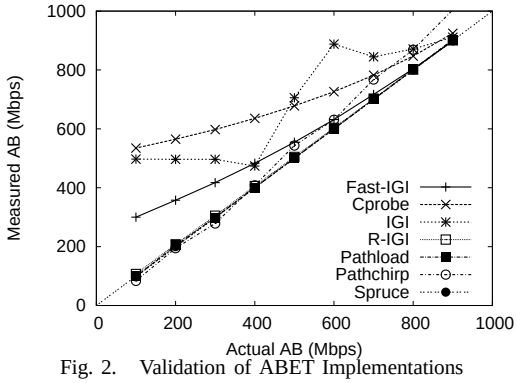


Fig. 2. Validation of ABET Implementations

Figure 2 plots the average of the estimated AB against the actual AB. We find that Pathload and Spruce are quite accurate in reporting the AB. PathChirp estimates deviate slightly at higher values of AB—we run the same set of experiments using a publicly-available NS-2 implementation of PathChirp and find that the AB estimates are quite similar to our implementation. CProbe, IGI, and Fast-IGI do a poor job of estimating the AB in the high-speed setting simulated. Cprobe works on the simple logic of sending a stream of packets at a fairly high rate (given by the bottleneck capacity)—the rate at which the probe-stream arrives at the receiver is taken as the estimate of the end-to-end AB. It has been shown in [22] that the receiving rate in such cases is not a good estimate of AB. Due to its inaccuracy, we do not use Cprobe for our subsequent evaluations.

We next investigated the reasons for the poor performance of IGI. Note that IGI always over-estimates the AB. On careful examination of the IGI design and implementation, we discovered a key design factor that was leading to over-estimation on high-speed network paths: The equation used for estimating the cross-traffic load (Eq. 3 in [23]) uses the link capacity as the multiplier—in our understanding, it should be using the current sending rate as the multiplier. We change our IGI implementation accordingly to create a new version, henceforth referred to as R-IGI. Figure 2 also plots the results of R-IGI validation—we find that R-IGI performs quite well. In our subsequent experiments, we use R-IGI.

Our implementation of Fast-IGI (validated in Fig 2) also incorporates the above-mentioned changes. However, it still leads to high estimation errors when the traffic load is higher than 500 Mbps. Since most of our subsequent evaluations are not conducted at such high loads, we include Fast-IGI in our subsequent evaluations.

V. Evaluating the Accuracy of ABETs in Dynamic Traffic Conditions

The validation experiments presented in Section IV also confirm the high accuracy of several prominent ABETs when the network traffic load does not change. In reality, this is

[6] Unless stated otherwise, all link capacities and link delays in all of our topologies are set to $1Gbps$ and $10ms$, respectively, and sufficient buffers are provisioned to avoid packet losses.

seldom the case with loaded Internet links. In order to reproduce in our simulations, the dynamic traffic conditions that characterize real Internet links, we rely on *replaying* packet-level traces collected from several Internet links. Specifically, we collect five 1-hour packet traces (from four different Internet links) which are summarized in Table II—the traffic load of these traces ranges from $160Mbps$ to $530Mbps$. We then use the *replay trace* module in NS-2 for creating an exact replica of the link-level packet-arrival process (and consequently, the AB process) for each trace. In this section, we evaluate ABET accuracy against this type of cross-traffic.

Trace	Traffic Type	Average Load
Ibiblio	Web server access link	160 Mbps
UNC05	University access link	230 Mbps
UNC28	University access link	358 Mbps
IPLS-CLEV	Internet2 backbone link	410 Mbps
IPLS-KSCY	Internet2 backbone link	530 Mbps

TABLE II

TRACES USED FOR EVALUATIONS

A. Single Bottleneck Scenario

We first evaluate the tools using the topology of Fig 1, but with the traces replayed (instead of CBR traffic) as cross-traffic between nodes N2 and N3. This topology represents paths on which an ABET is likely to encounter only a single congested link. We use this setup to study the impact of traffic load, MT, SI, and RTT on the AB estimation accuracy of different tools.

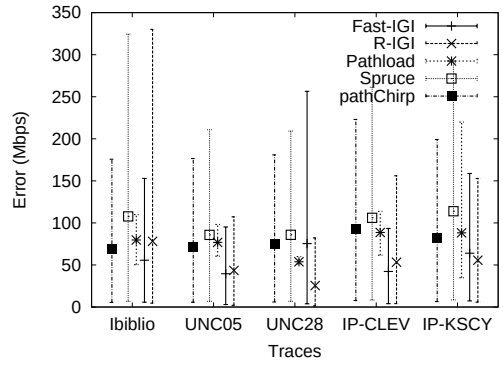


Fig. 3. Tool errors with default parameters

1) Default Tool Configuration: We first run each ABET against all five traces, using the default configuration of tool parameters, which dictate the implicit choices of MT and SI—the default MT for Pathload, PathChirp, R-IGI, Fast-IGI, and Spruce are roughly: 10 ms, 10 ms, 1 ms, 1 ms, 0.5 ms, respectively, and the default SI for both Spruce and PathChirp is 0.1. Each tool is run back-to-back for 300 seconds and the AB estimation error of each run is computed. Fig 3 plots the average, and the 5- and 95-percentiles (as error bars), of this estimation error for each tool and trace used. We observe that:

- The average estimation errors of ABETs are higher with dynamic cross-traffic than with CBR cross-traffic (Section IV), and range from 20 - 120 Mbps. Pathload, PathChirp, and Fast-IGI have similar average estimation errors, while R-IGI has lower and Spruce has higher errors.
- The estimation errors vary widely around the average. The variability is least for Pathload and quite high for Spruce and PathChirp—estimation errors sometimes exceed 300 Mbps.
- For each tool, the AB estimation errors are similar across the five traces, even though the traffic load in these traces are quite different. However, it is important to remember that the highest link utilization represented by these traces is only 53%—it is not clear if higher loads would impact estimation errors.

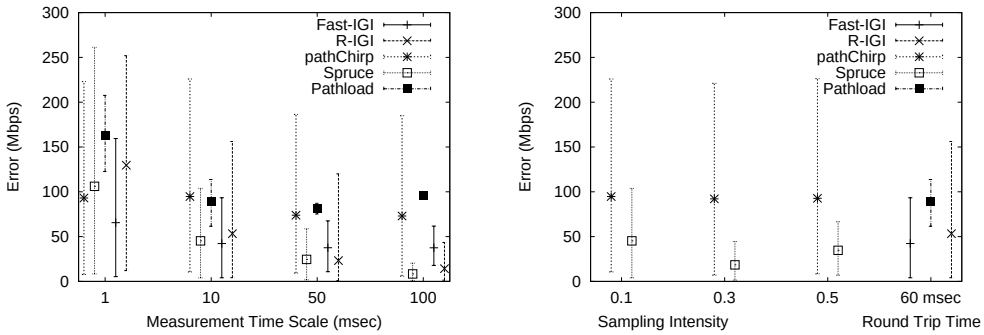
2) Impact of MT, SI, and RTT: The default choices of MT and SI vary widely across existing ABETs [20]. In order to compare tool performance in an unbiased manner, we next systematically control MT, SI, and RTT, and study the impact on the AB estimation error of each ABET. Specifically, we select MT from (1, 10, 50, 100 ms), SI from (0.1, 0.3, 0.5) for open-loop tools, and RTT from 60-300 ms for closed-loop tools—these values are representative of the diversity found in existing ABETs and Internet paths [20], [24].

Fig 4 plots the average and 5- and 95- percentiles of the AB estimation error with the IPLS-CLEV trace—the trends are quite similar for the other traces and are omitted due to space constraints. We observe that:

- Increasing the MT improves the accuracy of all ABETs. This is to be expected—larger MTs imply that a larger number of probe packets interact with the cross-traffic and are able to better sample the AB process. However, the gain in accuracy is most significant at fine time-scales. The gains are negligible beyond an MT of 50 ms.
  The impact of MT on PathChirp is lower than on the other tools. This is due to the exponential inter-packet spacing in the probe streams—the number of probes sent does not increase proportionally with MT.
- More importantly, by keeping the MT the same across different tools, the relative performance difference between the tools changes! Most significantly, Spruce now is the most accurate, while it was the least accurate with the default settings of MT.
- SI has a negligible impact on the AB estimation accuracy of the open-loop tools, Spruce and PathChirp. This result may seem contrary to the observations made in [20] that high values of SI lead to better sampling accuracy—it is important to note, however, that the AB estimation accuracy is also limited by the accuracy of the inference logic used by the respective tools. Our observations indicate that increasing the rate of probing the AB process is not likely to help improve the accuracy of current tools.
- Similar to SI, RTT has no impact on the AB estimation accuracy of the closed-loop tools, Pathload and R-IGI—we omit the detailed plots due to space constraints.

B. Multiple Bottlenecks

1) Bottleneck Location—Different Tight and Narrow Links: The inference logic of several ABETs—including IGI and Spruce—is based on the premise that on the path for which AB is to be estimated, the tight as well as narrow link are the same. In practice, this may not be the case with many Internet paths—indeed, an ISP access link that is shared among a



(a)IPLS-CLEV: Impact of MT (SI=0.1, RTT=60 ms) (b)IPLS-CLEV: Impact of SI (MT=10 ms)

Fig. 4. Impact of MT, SI, and RTT

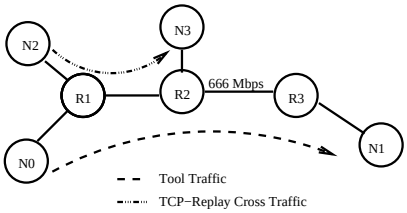


Fig. 5. Different tight and narrow links

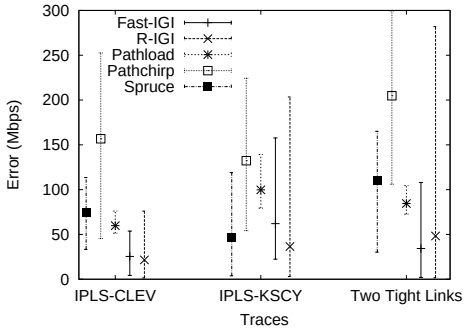


Fig. 6. Performance with multiple bottleneck links (MT=50ms, SI=0.1)

large user population may have a lower AB than the last-mile narrow link for many broadband users. In order to study ABET performance on such paths, we simulate the topology of Fig 5. The 666 Mbps link between the routers R2 and R3 is the narrow link (all other links have a 1 Gbps capacity). The ABETs run between the nodes N0 and N1. We replay traces between nodes N2 and N3 in order to ensure that link R1-R2 is the tight link for the tool traffic—for this, we use two traces: IPLS-CLEV (410 Mbps) and IPLS-KSCY (530 Mbps). We compute the actual end-to-end AB in any given time interval as the *minimum* of the AB on links R1-R2 and R2-R3. We use this to compute the AB estimation error for each tool run. Fig 6 plots the average and the 5- and 95-percentiles of the AB estimation errors observed from several back-to-back tool runs, with MT of 50 ms, and SI of 0.1. We observe that the error of PathChirp and Spruce increases by a factor of 2-3, compared to the scenario of Fig 1, while the performance of other ABETs is not impacted much. With this change, the relative rankings of Spruce and PathChirp changes and these now have the highest estimation errors.

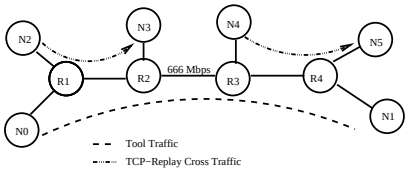


Fig. 7. Single narrow link; two tight links

2) Multiple Bottleneck—Two Potential Tight Links: Most ABET designers implicitly (and often, explicitly) assume the existence of only a *single* congested (bottleneck) link on the concerned path. It is conjectured that current ABETs might underestimate end-to-end AB in the presence of multiple bottleneck links [25]. In order to study this scenario, we simulate the topology of Fig 7. We replay the IPLS-CLEV trace between N2 and N3, and the IPLS-KSCY trace between nodes N4 and N5. The ABETs are run between nodes N0 and N1. With this setup, the tools encounter one narrow link (R2-R3), and two potential tight links (R1-R2 and R3-R4)—on an average, the latter is the "tighter" link; however, the tool traffic experiences queuing at both links.

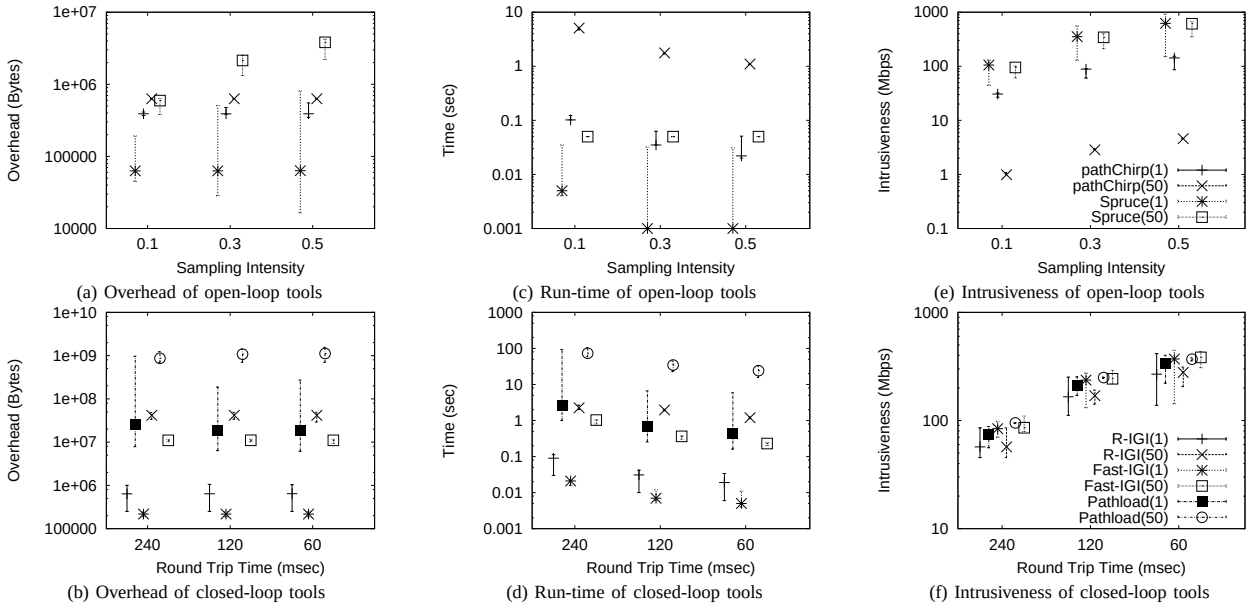
We compute the actual end-to-end AB as the *minimum* of the AB on links R1-R2 and R3-R4. We run each ABET several times with MT of 50 ms and SI of 0.1. Fig 6 plots the average and the 5- and 95-percentiles of the AB estimation errors of different tools. On comparison to the other plots on the same figure, we observe that the accuracy of Pathload, R-IGI, and Fast-IGI is not significantly impacted by the presence of multiple tight links. However, the accuracy of PathChirp and Spruce further degrades and these are the most inaccurate.

VI. Evaluating the Costs of ABETs

A. Quantifying tool costs

Our evaluation so far considers only the AB estimation accuracy of ABETs. We next quantify the cost of measuring AB by computing the *overhead*, *run-time*, and *intrusiveness* (defined in Section III) of each tool run for all experiments described so far. In the interest of space, we present results only for the topology of Fig 1—the trends are similar for the other topologies and traces.

a) Overhead: Fig 8 (a), (b) plot the average and the 5- and 95-percentiles of the overhead for each tool at different SI and with MT of 1 and 50 ms. We observe that:



(a) Overhead of open-loop tools (c) Run-time of open-loop tools (e) Intrusiveness of open-loop tools

(b) Overhead of closed-loop tools (d) Run-time of closed-loop tools (f) Intrusiveness of closed-loop tools

Fig. 8. Costs of ABETs with the Ibiblio trace —refer to (e) and (f) for legends (numbers in parenthesis indicate the MT in ms).

- For any given MT, PathChirp, R-IGI, and Fast-IGI have the least overhead. The overhead of each run of Pathload is larger by more than an order of magnitude and can be as high as a giga-byte.
- Tool overhead increases with MT. While the increase is linear for most tools, it is not for PathChirp. This is because PathChirp uses an exponentially-spaced packet stream—increasing the stream duration, therefore, increases the number of packets only sub-linearly. Consequently, while the overhead of Pathload increases from 50 MB to 2.5 GB as the MT increases from 1 to 50 ms, the overhead of PathChirp increases from 0.5 MB to only 0.75 MB.
- SI and RTT have no impact on the overhead of most tools. The overhead is dictated by the size and number of probe streams sent—the same number of probe-streams are needed to arrive at an AB estimate, irrespective of these quantities. For Spruce, however, the overhead increases with SI—this is an artifact of the fact that we are using a large number of tool runs in order to incorporate SI into its AB estimates.

b) Run-time: Fig 8 (c), (d) plot the average and the 5- and 95-percentiles of the run-time for each tool at different SI and with MT of 1 and 50 ms. We observe that:

- Spruce is the fastest tool; this is true in spite of the fact that we aggregate several tool-runs in order to get an estimate at the desired MT.
  Pathload is the slowest tool, taking 10-100 seconds to return an AB estimate. R-IGI takes 1-10 seconds and PathChirp takes a few seconds (with its typically configured MT). Fast-IGI is roughly 5 times faster than R-IGI.
- Increasing the MT results in a proportional increase in the run-time of all tools. For Spruce, however, this is an artifact of the way we are incorporating MT into the AB estimates yielded by it.
- The run-time of closed-loop tools is proportional to the path RTT, which characterizes the feedback delay of these tools.
- As SI increases, the run-time for PathChirp decreases. This is because open-loop tools such as PathChirp rely on sending and observing a fixed number of probe streams—larger is the SI, faster would these streams be sent.
  SI has no impact on the run-time for Spruce—this is, however, an artifact of the way we incorporate MT into the Spruce estimates.
- The run-times of most tools are predictable—they do not vary significantly around the average.

c) Intrusiveness: Fig 8 (e), (f) plot the average and the 5- and 95-percentiles of the intrusiveness for each tool at different SI and with MT of 1 and 50 ms. Recall that intrusiveness is given by the ratio of the overhead to run-time of a tool. We observe that:

- All closed-loop tools are quite intrusive and can temporarily congested high-speed links. The run-times suggest that tools like Pathload can induce such congestion for several seconds.
  Spruce is also quite intrusive—it sends back-to-back probe packets at the line rate. However, since its run-time is small, it is unlikely to induce congestion for long durations (unless it is run several times). PathChirp is the most non-intrusive tool.
- The closed-loop tools—Pathload, R-IGI, and Fast-IGI—have very similar intrusiveness. This may seem surprising given that both the run-times and overheads of these tools vary by nearly two orders of magnitude. However, it is important to note that all of these tools rely on a feedback

loop and iteratively search for the AB—consequently, these operate at time-units that are RTT long. Both R-IGI and Pathload use the concept of self-loading streams and, consequently, their per-RTT overhead (which is the intrusiveness) is similar.

- As MT increases, the intrusiveness of closed-loop tools increases proportionally. This is to be expected; the per-RTT overhead of these tools is given by the size of each probe-stream, which is proportional to the MT.
  Increasing MT decreases the intrusiveness of PathChirp. As mentioned before, while the run-time of PathChirp increases linearly with MT , its overhead increases only sub-linearly due to the exponential nature of the probe stream. The intrusiveness, consequently, decreases.
  MT has no impact on Spruce—however, this is also an artifact of the way we incorporate MT into its AB estimates.
- Increasing SI increases the intrusiveness of open-loop tools. This is to be expected, as a larger number of probe streams are sent per unit time as a result of increasing SI.
- Intrusiveness of closed-loop tools increases as the RTT decreases. This is to be expected as the per-RTT overhead remains the same.

We conclude that in terms of the cost metrics, the tool that is likely to run quickly, while not perturbing ongoing traffic much seems to be PathChirp; the cost of Pathload, R-IGI, and Fast-IGI seems to be the highest.

B. Impact on Responsive Cross-Traffic

TCP is the dominant transport protocol used by most Internet applications [26]. TCP uses congestion-control mechanisms to reduce the data sending rate on detecting network congestion. A key issue in the wide-scale deployment of ABETs is that of how adversely do these tools impact the performance of applications that rely on such *responsive* transport protocols. In this section, we study this issue.

Unfortunately, the trace replay methodology used in Sections V and VI-A is not suitable for studying this issue—it recreates only the link-level packet-arrival process and does not incorporate TCP behavior. In particular, it does not model the impact of queuing delays and losses on the subsequent packet transmission behavior of a TCP connection. Recent efforts have focused on developing traffic-generation tools that also incorporate the responsive behavior of TCP—Tmix [27] is one such tool. It takes as input a link-level packet trace (such as those summarized in Table II), and for each TCP connections that appears in the trace, it derives the RTT and the application-level data generation behavior (including user think times). Recently, an NS-2 version of Tmix has been developed [28], which takes this derived connection descriptor as input and emulates per-connection application bots with similar RTTs and data-generation behaviors.

We use this version of Tmix in the topology of Fig 1, in which the nodes N2 and N3 now each emulate a cloud of servers and clients that instantiate connections between these two nodes. Different per-connection RTTs are simulated using the delay-box environment [29] of NS-2 and the router buffer sizes are limited to 100 MSS-sized packets to help emulate packet losses (even without the ABETs). The connections to be simulated are derived from a real Internet access link and have an average traffic load of 300 Mbps.

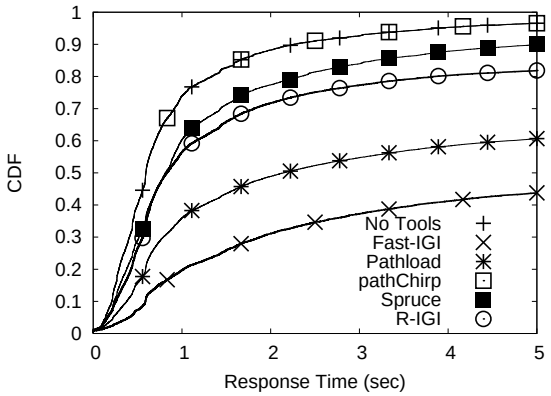


Fig. 9. CDF of response times with default parameters

Tmix measures and reports the per-connection response time—the time taken for the connection to transfer all data between the two end-points. In order to assess the impact of ABETs on the simulated TCP connections, we run the tools continuously between nodes N0 and N1 and observe the impact on the distribution of connection response times. Each tool has an SI of 0.1, an MT of 50ms, and an RTT of 240 ms. Fig 9 plots this distribution for experiments conducted with each tool and without any tools. We find that:

- PathChirp has no noticeable impact on connection response times. As seen in Section VI-A, PathChirp has a fairly low intrusiveness—it does not cause much queue build-up on the bottleneck link.
- All of the other tools can significantly impact the response times of TCP connections. Of these, Spruce and R-IGI increase the connection response times for long connections by a factor of 2-3. For instance, the 75-percentile response time without any tool is a little over 1 s, while with Spruce and R-IGI , the 75-percentile response times are 1.7 s and 2.5 s, respectively.
  Note that while Spruce had one of the highest measures of intrusiveness, it does not fare among the worst in impacting connection response times. Its packet steams are too small (two packets) to sustain congestion long enough to inflict packet losses.
- Pathload and Fast-IGI can significantly impact the response times of all connections. While 75% of connections have a response time less than 1 s in the absence of any tool, nearly 65% and 80% of connections have a response time larger than 1 s in the presence of Pathload and Fast-IGI, respectively.

We conclude that is an application needs to run an ABET repeatedly on a given Internet path, it should use PathChirp.

Such an application should never use Pathload or Fast-IGI as these are likely to significantly impact connection response times.

VII. Concluding Remarks

In this paper, we conduct a comprehensive empirical evaluation of existing algorithmic techniques used for measuring end-to-end AB. We study both the accuracy and costs of deploying ABETs. Our study reveals several insights into the relative performance of ABETs, in the absence of limitations of implementation technology. A salient feature of our study is that we incorporate the impact of two key probing parameters, the measurement timescale and the sampling intensity.

Some of our key observations are:

- Modifying existing ABETs to allow the choice of MT and SI is not straightforward (and is sometimes not even possible). Given the impact of these quantities on AB estimation, it is important for new ABETs to explicitly incorporate these.
- The accuracy of most ABETs can be improved by using an MT of 50 ms. In particular, Spruce uses a fairly small MT by default—the ranking of Spruce changed from the least accurate tool to the most accurate tool when its MT was made comparable to that of other tools.
  SI and path RTT have negligible impact on the accuracy of existing tools.
- While Spruce is the among the most accurate ABETs for paths with a single bottleneck link, its accuracy worsens for paths with multiple bottleneck links.
- PathChirp has the lowest overhead among existing ABETs, especially for large values of MT. Even when PathChirp is run continuously on a path, it has virtually no impact on the response times of TCP connections sharing the same path.
  Spruce is the fastest tool, but has the highest value of intrusiveness. But it does not adversely impact of the response times of responsive TCP connections to the same extent as the other tools. Pathload and Fast-IGI can significantly worsen the response times of TCP connections, if these are run repeatedly on a path.

References

[1] C. Dovrolis, R. Prasad, and M. Jain, "Socket Buffer Auto-Sizing for High-Performance Data Transfers," *Journal of Grid Computing*, vol. 1(4), 2004.

[2] R. L. Carter and M. E. Crovella, "On the network impact of dynamic server selection," *Computer Networks*, vol. 31, no. 23-24, pp. 2529–2558, 1999.

[3] A. Shriram and J. Kaur, "Identifying bottleneck links using distributed end-to-end available bandwidth measurements," *First ISMA Bandwidth Estimation Workshop*, December 2003.

[4] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE JSAC Internet and WWW Measurement, Mapping, and Modeling*, 2003.

[5] M. Jain and C. Dovrolis, "Pathload: an available bandwidth estimation tool," in *PAM*, 2002.

[6] V. Ribeiro, "pathChirp: Effi cient Available Bandwidth Estimation for Network Path," in *PAM*, 2003.

[7] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference '03*, Miami, Florida, October 2003.

[8] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks, Tech. Rep. 1996-006, 15, 1996. [Online]. Available: citeseer.ist.psu.edu/carter96measuring.html

[9] N. Hu and P. Steenkiste, "Estimating available bandwidth using packet pair probing," 2002. [Online]. Available: citeseer.ist.psu.edu/hu02estimating.html

[10] G. Jin, "netest-2," 2004, http://www-didc.lbl.gov/NCS/netest.html.

[11] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Global Internet Symposium*, November 2000.

[12] J. Navratil, "ABwE: A Practical Approach to Available Bandwidth," in *PAM*, 2003.

[13] J. Alberi, A. McIntosh, M. Pucci, and T. Raleigh, "Overcoming precision limitations in adaptive bandwidth measurements," in *3rd New York Metro Area Networking Workshop (NYMAN)*, September 2003.

[14] G. Jin and B. Tierney, "System capability effects on algorithms for network bandwidth measurement," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2003.

[15] A. Pasztor and D. Veitch, "Pc based precision timing without gps," in *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM Press, 2002, pp. 1–10.

[16] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and K. C. Claffy, "Comparison of public end-to-end bandwidth estimation tools on high-speed links." in *PAM*, 2005, pp. 306–320.

[17] F. Coccetti and R. Percacci, "Bandwidth measurements and router queues," Instituto Nazionale Di Fisica Nucleare, Trieste, Italy, Tech. Rep. INFN/Code-20 settembre 2002, 2002, http://ipm.mib.infn.it/bandwidth-measurements-and-router-queues.pdf.

[18] F. Monetesino-Pouzols, "Comparative analysis of active bandwidth estimation tools," in *Proceedings of Passive and Active Measurement Workshop (PAM)*, April 2004.

[19] X. Liu, K. Ravindran, and D. Loguinov, "Evaluating the potential of bandwidth estimators," in *4th New York Metro Area Networking Workshop (NYMAN)*, September 2004.

[20] A. Shriram and J. Kaur, "Empirical study of the impact of sampling timescales and strategies on measurement of available bandwidth," in *Proceedings of Passive and Active Measurement Workshop (PAM)*, March 2006.

[21] "Network simulator-2 ns2 (http://www.isi.edu/nsnam/ns/)."

[22] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, October 2004.

[23] N. Hu and P. Steenkiste, " Evaluation and characterization of available bandwidth probing techniques," in *IEEE Journal on Selected Areas in Communications*, Aug 2003.

[24] J. Aikat, J. Kaur, D. Smith, and K. Jeffay, "Variability in TCP round-trip times," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2003.

[25] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," 2002. [Online]. Available: citeseer.ist.psu.edu/jain02pathload.html

[26] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffi c patterns and characteristics," *IEEE Networks*, vol. November/December, 1997.

[27] F. Hernandez-Campos, F. D. Smith, and K. Jeffay, "Generating realistic tcp workloads," *Proceedings of CMG*, pp. 273–284, December 2004.

[28] M. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, and F. Smith, "Tmix: A tool for generating realistic application workloads in ns-2," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 67–76, 2006.

[29] "Per-fbw delay and loss in ns-2 with delaybox," http://dirt.cs.unc.edu/delaybox/.