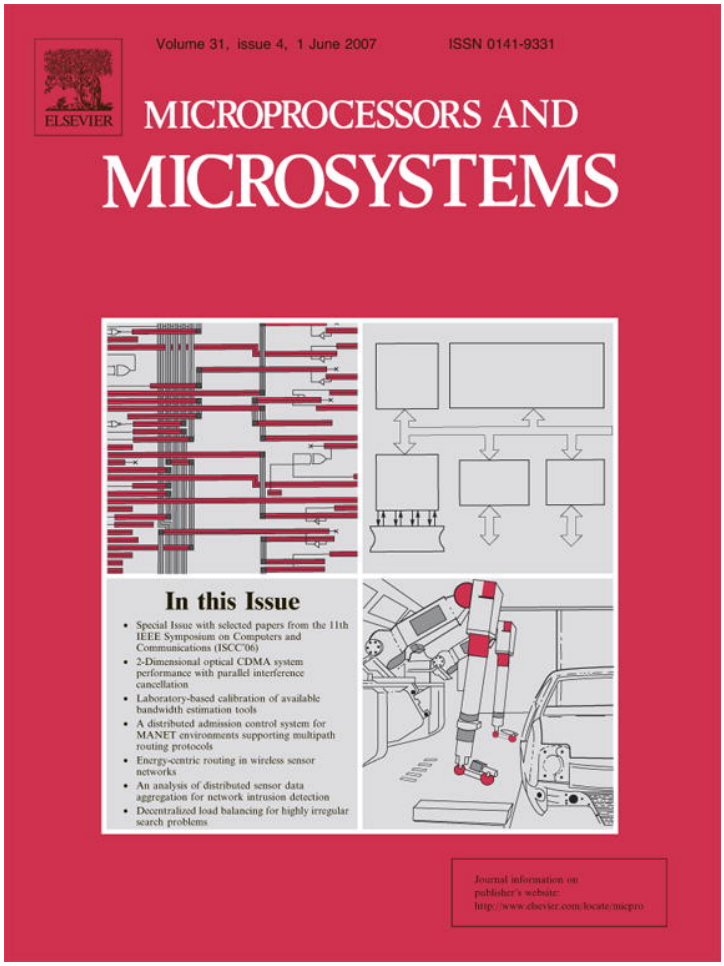

Available online at www.sciencedirect.com
ScienceDirect
Microprocessors and Microsystems 31 (2007) 222–235
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Laboratory-based calibration of available bandwidth estimation tools

Joel Sommers [a,*], Paul Barford [a], Walter Willinger [b]

[a] University of Wisconsin-Madison, Madison, WI, USA
[b] AT&T Labs-Research, Florham Park, NJ, USA

Available online 13 January 2007

Abstract

Examining the validity or accuracy of proposed available bandwidth estimation tools remains a challenging problem. A common approach consists of evaluating a newly developed tool using a combination of simple ns-type simulations and feasible experiments *in situ* (i.e., using parts of the actual Internet). In this paper, we argue that this strategy tends to fall short of establishing a reliable "ground truth," and we advocate an alternative *in vitro*-like methodology for calibrating available bandwidth estimation tools that has not been widely used in this context. Our approach relies on performing controlled laboratory experiments and using a set of tools to visualize and analyze the relevant tool-specific traffic dynamics. We present a case study of how two canonical available bandwidth estimation tools, SPRUCE and PATHLOAD, respond to increasingly more complex cross traffic and network path conditions. We expose measurement bias and algorithmic omissions that lead to poor tool calibration. As a result of this evaluation, we designed a calibrated available bandwidth estimation tool called YAZ that builds on the insights of PATHLOAD. We show that in head to head comparisons with SPRUCE and PATHLOAD, YAZ is significantly and consistently more accurate with respect to ground truth, and reports results more quickly with a small number of probes.

Keywords: Available bandwidth; Active measurement; Calibration; YAZ

1. Introduction

Calibration strategies for Internet measurement tools are essential for detecting inaccuracy in the underlying data, and misconceptions or errors in their analysis [1]. In this paper, we propose and investigate a set of calibration techniques that can greatly increase our confidence in the validity and accuracy of end-to-end available bandwidth estimation tools (ABETs). Echoing the same sentiment as expressed by Paxson in [1], tool calibration is not meant to achieve perfection. Rather, it is to aid in our understanding of the tools and their applicability by producing results that are close to the "ground truth." Calibration may also illuminate the circumstances under which the tools may give inaccurate results.

There are two conventional and complementary aspects to calibration: *comparison* with a known standard, and (if necessary) *adjustment* to match a known standard. The first notion encompasses the task of comparing the output of a measurement tool with "ground truth" – a known quantity like the reading of an accurate and precise device. For ABETs, this activity involves comparison with measurements of available bandwidth (AB) that have been obtained through, e.g., packet traces with timestamps of sufficient quality. The second facet of calibration involves changing some feature of a measurement tool so its output matches a standard as closely as possible. In the context of ABETs, this aspect of calibration may involve adjusting parameters of a given algorithm, or the algorithm itself.

Traditional approaches for calibrating and validating ABETs almost always employ two basic strategies. One is the use of simple ns-type simulations, and the second consists of small-scale experiments in the "wild," i.e., largely uncontrolled tests that use a small set of paths in the live Internet. Ns-type simulations are attractive since they have

* Corresponding author.
E-mail addresses: jsommers@cs.wisc.edu (J. Sommers), pb@cs.wisc.edu (P. Barford), walter@research.att.com (W. Willinger).

0141-9331/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2006.12.004

the advantage of simplified implementations and complete experimental control. However, by definition they are an abstraction of networking reality [2,3] which may render their results largely irrelevant in situations when the details of live system and protocol implementations or traffic conditions have little in common with their simulation-based counterparts. In contrast, experiments that use paths in the live Internet encounter networking systems, protocols and traffic conditions (depending on the part of the Internet to which they are confined) similar to what would be expected in other paths of the network. However, experiments run in the wide area are largely uncontrolled and typically lack the necessary instrumentation for establishing a reliable standard against which results can be compared and understood. While networking researchers have been generally aware of the pros and cons of these two strategies, the lack of realism in ns-type simulations and the lack of control and instrumentation in the wide area cast serious doubts on these predominant approaches to ABET calibration and validation, and highlight the need for improved calibration strategies.

In this paper, we investigate an alternative ABET calibration strategy based on conducting experiments in a laboratory setting that is amenable to establishing the "ground truth" for a great variety of Internet-like scenarios. This setting should include, wherever possible, the use of actual hardware found on end-to-end paths in the Internet (e.g., routers, switches, etc.), the use of various versions of the full TCP/IP protocol stack, workload generators capable of exercising systems over a range of realistic conditions, and measurement devices that provide a level of accuracy suitable for establishing ground truth. By advocating such an *in vitro*-like experimental environment, we combine the advantages of ns-type simulations (i.e., complete control and full instrumentation) with those offered by experiments in the wide area (i.e., more realistic network systems, protocols and traffic dynamics). Laboratory-based calibration techniques are established in other scientific disciplines such as chemistry and biology but they have not seen widespread application to network measurement tools. While the focus of this paper is on a calibration strategy in the context of ABETs, our future plans include investigating generalizations to our approach to additional active measurement-based tools that attempt to infer network-internal characteristics.

Estimating the AB along a network path is a topic that has received considerable attention in recent years [4–13]. Informally, end-to-end available bandwidth (AB) is defined as the minimum spare capacity on an end-to-end path between a sender and receiver. To calibrate and validate ABETs, a detailed understanding of realistic queuing effects experienced by individual packets as they compete and interact with other packets is essential, and requires fine-grained, time-synchronized measurements of packets as they arrive at *and* subsequently depart from the different routers along the network path.

We address the need for analyzing such detailed measurements from an instrumented laboratory setting with multiple tools, including *phase plot analysis* [14]. Phase plots enable visualization and analysis of a variety of complex network traffic behavior at different levels of granularity (e.g., from packets to flows to more application-specific quantities) and over a range of time scales. Our approach to phase plot analysis is based on taking measurements of *the spacings between consecutive packets* in individual packet streams as they enter a router and then as they exit that same router. We use these measurements as the x- and y-coordinates of points in the 2D ingress–egress space. The resulting phase plots enable a transform-free representation of how the spacings between individual packets change as these packets pass through a router. This simple representation can provide surprisingly rich insights into a wide variety of network behavior. Within the context of ABETs, we advocate phase plot analysis as a natural means for assessing two characteristics critical to measurement tool calibration. The first is the ability to identify and isolate sources of bias in active measurement tools themselves. The second is the ability to assess the basic design assumptions of ABETs concerning the nature of complex queuing effects that individual packets or flows experience at routers as they traverse the network and interact with competing traffic.

Using an openly available laboratory testbed [15], we apply our calibration strategy through a series of experiments to two ABETs, SPRUCE [13] and PATHLOAD [7], which we consider to be canonical representatives of two basic methods for ABE. We analyze the detailed arrival and departure measurements available in our testbed, showing why and how both tools are prone to measurement bias and errors over a range of increasingly complex cross traffic and network path conditions. With the insights gained from analyzing the detailed arrival and departure measurements, we designed and implemented a calibrated ABET, called YAZ, that builds on the basic insights of PATHLOAD. Through an additional set of laboratory-based calibration tests, we show that (1) YAZ compares well with respect to known measures of AB, (2) it is significantly more accurate than both SPRUCE and PATHLOAD, while remaining much less intrusive than PATHLOAD, and (3) it produces available bandwidth estimates faster than the other tools. If PATHLOAD and SPRUCE represent a tradeoff between measurement overhead and measurement accuracy, our results for YAZ show that this tradeoff is not fundamental. For a SPRUCE-like budget, YAZ is more accurate than PATHLOAD, sometimes significantly so.

2. Background and related work

Dynamic estimation of end-to-end available bandwidth (spare capacity) has important potential for network capacity planning and network overlay monitoring and management. Active measurement tools for estimating or inferring AB are designed to send precisely crafted packet

pairs or streams and – by measuring perturbations of the pairs or streams as observed at a receiver – to infer the bandwidth available along a given end-to-end path. While the development of fast and accurate ABETs is an active area of research (see for example [4–13]), two recent tools, PATHLOAD [7] and SPRUCE [13], represent the two most common strategies for probing and two appealing methods for AB inference. Thus, these tools are the focus of our ABET calibration study. A study related to ours, but with a focus on evaluating ABETs in the context of high-speed links is found in [12]. Additional contrasts are that our focus is calibration (and thus the *mechanisms* that lead to performance differences between tools), not simply a head-to-head comparison of ABETs, that we use a much more varied range of cross traffic, and that our experimental environment is openly available. Furthermore, we used significantly more precise measurements than router SNMP counters and we created a new highly accurate and low-impact ABET as a result of this work.

2.1. Definitions and overview of ABE techniques

The *available bandwidth*, A, of a single link is defined as the amount of spare capacity over a given time interval, τ. If C is the link capacity and $U(x)$ gives the instantaneous link utilization (0 or 1) at time x, then the available bandwidth over the time interval $[t, t+\tau]$ is

$$A = C\left(1 - \frac{1}{\tau}\int_{t}^{t+\tau} U(x)\mathrm{d}x\right). \quad (1)$$

The end-to-end available bandwidth is then defined as the minimum AB over all hops in the path:

$$A \equiv \min\{A_i : i \in 1, \ldots, H\}, \quad (2)$$

where H is the path length in (layer 3) hops.

The link with smallest AB is referred to as the *tight link*, while the link with smallest capacity is called the *narrow link*. These definitions avoid the ambiguous term *bottleneck link* [7]. They also help to avoid any implicit assumption that the tight link is necessarily the narrow link. We use the term capacity in this paper to refer to the maximum transmission rate at layer 3, assuming a maximum frame size of 1500 bytes. For example, with standard Ethernet there is a nominal (physical layer) transmission rate of 10 Mb/s. However, for each packet delivered from layer 3 for transmission, there is an additional overhead of 38 bytes from layers 1 and 2. Thus, the transmission rate available to layer 3 is reduced to ≈9.75 Mb/s.

Existing tools for measuring AB are generally focused on either estimating the amount of cross traffic on the tight link, or on direct measurement of AB by sending probe streams at various rates. Both methods are accomplished by specifying an initial set of parameters, sending a series of probes and measuring responses, and then inferring an estimate of AB from the measurements. All methods for measuring available bandwidth assume, for simplification, a relatively homogeneous environment. First, they assume FIFO queuing at routers. Second, they assume that cross traffic is fluid (cross traffic packets are infinitely small). Finally, cross traffic intensity is assumed to be stationary over the measurement period.

2.2. SPRUCE

SPRUCE estimates AB by sending packet pairs spaced back-to-back according to the capacity C of the tight link.[1] Assuming fluid cross traffic, the amount by which the packet pairs are expanded by the tight link is proportional to the volume of cross traffic. If g_{in} is the spacing of back-to-back probe packets on the tight link and g_{out} the spacing measured at the receiver, the AB is calculated as:

$$A = C\left(1 - \frac{g_{\mathrm{out}} - g_{\mathrm{in}}}{g_{\mathrm{in}}}\right). \quad (3)$$

SPRUCE sends, by default, 100 packet pairs at Poisson-modulated intervals, and reports the average A over those samples.

2.2.1. SPRUCE limitations

The above formulation assumes that the tight-link capacity (C) is known *a priori*. Clearly, negative AB estimates are possible, although SPRUCE reports zero AB when an estimate is negative. Also, through the course of a measurement, it is unknown whether the estimate has yet converged to the true AB (assuming some stationary average exists).

Liu et al. [16] use a single-hop setting to analyze SPRUCE-like techniques and the bias introduced by the fluid assumption under a range of cross traffic conditions. They claim that bias is minimized when the input gap g_{in} is set to less than or equal to the back-to-back spacing on the tight link. However, a more complex topology with cross traffic on the non-tight links may still introduce bias.

There are practical limitations when considering high-bandwidth links. For the experiments discussed below, the fastest narrow link we consider is an OC-3 (155 Mb/s nominal transmission rate), which requires SPRUCE to send its packet pairs spaced by approximately 80 μs. While we will show that modestly provisioned hosts in our testbed can accommodate this requirement, emitting packet pairs to measure available bandwidth on OC-12 links (622 Mb/s nominal), where packets must be sent with spacing of 20 μs, borders on being infeasible with commodity workstations.[2]

[1] With SPRUCE, the tight link and narrow link are assumed to be the same. Strauss *et al.* claim that the estimates may still be meaningful even when this condition is not satisfied [13].

[2] Assuming frame sizes larger than 1500 bytes are not used. See Shriram et al. [12] for additional issues with ABE on high speed links.

2.3. PATHLOAD

PATHLOAD attempts to create short-lived congestion conditions in order to measure AB. It detects congestion through trends in one-way probe packet delays. Specifically, an increasing one-way delay (OWD) trend is assumed to be indicative of queueing and that the probe stream rate is greater than the end-to-end AB. Through iterative adaptation of its probe stream rate, PATHLOAD converges on an available bandwidth range, the center of which is the average spare capacity over the measurement period.

A simple way to describe the relationship between one-way delay trends and available bandwidth is in terms of probe stream rate. If L is the probe packet size (in bytes) and g_{in} the spacing between consecutive packets in a stream, the probe rate (input rate) produced by PATHLOAD is $r_{\text{in}} = \frac{L * 8}{g_{\text{in}}}$. An increasing one-way delay trend is equivalent to saying that there is an increasing inter-packet spacing trend, and an average increase in spacings causes the overall probe rate measured at the receiver (r_{out}) to be less than that introduced at the sender (r_{in}). Such a decrease is taken as evidence that the end-to-end AB is less than the probe stream rate. This relationship can be expressed as follows:

$$\frac{r_{\text{in}}}{r_{\text{out}}} = \begin{cases} \leqslant 1 & r_{\text{in}} \leqslant A \\ > 1 & r_{\text{in}} > A \end{cases} \tag{4}$$

PATHLOAD takes N measurements with probe streams of length K packets, iteratively adapting its send rate to determine whether or not there is an OWD trend. These N streams are referred to as a fleet. Each stream within a fleet is separated by an amount of time designed to allow the path to quiesce. By default, N is set to 12 and K to 100. After each of the N streams, PATHLOAD uses two separate metrics to estimate trends in OWD: the pair-wise comparison test (PCT) and the pair-wise difference test (PDT). The PCT and PDT metrics operate on blocks of $\Gamma = \sqrt{K}$ packets from each stream. For each block k, the median value, $\hat{D}^k$, is chosen as a robust estimator of OWD. These metrics are defined as follows:

$$A_{\text{PCT}} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}^k > \hat{D}^{k-1})}{\Gamma - 1}, \tag{5}$$

$$A_{\text{PDT}} = \frac{\hat{D}^{\Gamma} - \hat{D}^1}{\sum_{k=2}^{\Gamma} |\hat{D}^k - \hat{D}^{k-1}|}, \tag{6}$$

where $I(x)$ is a function returning 1 if the condition x is true, and 0 otherwise.

PCT returns a value between 0 and 1, where 0 indicates that the stream has not experienced an increasing OWD trend, and 1 indicates that, for $k \in [2, \ldots, \Gamma], \hat{D}^k > \hat{D}^{k-1}$; that is, there is a consistent increasing trend in OWD. PDT detects whether there is a net increasing trend, considering only the first and last OWD samples. It returns a value between -1 and 1, where -1 indicates a strongly decreasing OWD trend (i.e., the probe rate was measured to be much higher at the receiver than was sent), and 1 indicates a strongly increasing OWD trend. For each metric, a threshold value is chosen, above which the OWD trend is said to be increasing. These thresholds are 0.55 for PCT and 0.4 for PDT. The two metrics must agree (either increasing or non-increasing OWD) on a certain fraction of the streams in a fleet for an overall judgment to hold (by default, this threshold is 0.6). Otherwise, the result is inconclusive and additional fleets are required. The specific algorithm for how the input stream rate is modulated based on the outcome of the PCT and PDT metrics is described in [7] and in the source code to PATHLOAD.

2.3.1. PATHLOAD limitations

The PCT and PDT metrics, along with the specific thresholds, assume congestion only takes the form of *increasing* one-way delays, i.e., expansion in intra-stream packet spacings. In [10], Paxson notes that while expansion is the predominant result of queuing, compression events commonly occur. Moreover, compression is the result of packets earlier in a stream being held up, allowing subsequent packets to "catch up". Such a situation is indicative of a queue that is draining, which at least suggests that congestion existed in the very recent past. The PCT metric does not consider compression, and the default threshold used for the PDT metric eliminates compression as an indication of congestion.

Like SPRUCE, ABETs that use self-loading techniques may not be able to produce streams on commodity systems that are sufficient for detecting AB at OC-12 link speeds and above without using large frame sizes. Another potential problem is that congestion-inducing measurement tools may cause significant and/or persistent packet loss during use. To minimize impact, packet streams should be short and should be spaced far enough apart for cross traffic sources to recover from any losses. However, shorter streams result in higher measurement variance, as noted in [7,17]. An additional problem with longer streams is that there is an increased possibility for operating system context switches, causing a disruption in intended packet spacings and invalidating that stream. PATHLOAD requires threshold parameter selection in its OWD trend detection algorithm. To date, there has been no study of sensitivity of these metrics to different cross traffic, and default thresholds were selected based on experience with network simulations [7].

3. Calibration framework

Comparison with a standard and subsequent adjustment of an ABET's algorithm or parameters are complementary activities of calibration. The basic task of comparing the output of an ABET with the actual AB over a time interval requires relatively simple measurements. However, to gain

insight into *how* an ABET arrives at a particular estimate we require measurements and analysis suited to the probes produced by an ABET and the reported measurements. We also require appropriate test environments to evaluate ABET accuracy over a range of controlled conditions and to expose algorithmic or parametric assumptions that may need adjustment.

As part of our framework, we offer a set of issues to consider for ABET calibration:

(1) There are performance and predictability limitations imposed by the operating system (OS) and hardware (e.g., general purpose workstations with standard network interface cards) running the measurement tools. Two key considerations are whether probe packet streams (specifically spacing between packets) can be generated with sufficient fidelity, and if time stamp accuracy (and in some cases, synchronization) is sufficient.
(2) Assumptions about and/or abstract models for the behavior of routers and switches in networks are the foundation for inference methods used to interpret active measurements. The diversity of the implementation details of those systems can limit the effectiveness of the inference methods.
(3) Probes and response packets generated during active measurement impose a load on the network which can change the conditions on the path of interest and potentially skew results.
(4) The heterogeneity and burstiness of traffic can extend beyond the operating bounds of the tool.
(5) Many active probe tools require specification of a set of parameters before they are used. A tool's effectiveness can be limited by its sensitivity to configuration parameters.

The first two issues imply that certain assumptions, while valid in simulation, may lead to unexpected behavior when an ABET is deployed in live Internet environments. Issues 3 and 4 imply that fully instrumented environments are key for understanding the impact and reported measurements of ABETs. The final issue identified above suggests that tool calibration should be performed in a controlled, yet realistic (to the extend possible) environment.

3.1. Calibration strategy

To address the above issues, we advocate the use of laboratory-based testbeds for calibrating ABETs. Such environments provide an important set of capabilities not offered by standard simulation [18] or *in situ* settings such as PlanetLab [19], including repeatability, transparency, and the use of actual systems and implementations of actual protocols. The essence of our calibration strategy for this study consists of the following.

(1) Design appropriate test environments where a standard can be established over a range of increasingly complex, repeatable test conditions. Essential to thi sfirst step is the availability of hardware that provides measurements with a level of accuracy greater than the ABET. Such accuracy is typically not available for *in situ* studies.
(2) For the setups defined in the first step, identify relevant test suites for assessing issues such as host system capabilities, loading effects, and network system behavior over a range of expected conditions. Real systems are generally required to study such issues.
(3) The evaluation of data collected in the testbed should be aided by flexible analysis and visualization techniques that provide insight into relevant traffic dynamics and, ultimately, the available bandwidth process that the ABET attempting to measure or infer.

Availability of open lab-based environments that deploy general-purpose workstations and network systems is on the rise [15,20,21]. Although similar environments have been used successfully in recent studies [12,22], they are not openly available to external researchers, and have seen little use for *calibration* of ABETs. A possible concern in this regard is the ability to conduct tests with "representative" traffic conditions in a lab environment. However, tools such as [23–25] have addressed this problem to some extent.

3.2. Calibration measurements and analysis

The interactions of ABET measurement probe packets with cross traffic in the different routers along the end-to-end path occurs on time scales that are typically in the range of tens to hundreds of microseconds. To gain insight into ABET behavior, we capture time-synchronized packet headers before *and* after interaction between probes and cross traffic at a congested queue. From these measurements, we can compare how packet streams intended by an ABET differ from the stream actually produced. From the arrival measurements, we can construct a true measure of AB over a given time interval as a standard by which to judge an ABET.

To analyze the probe arrival (ingress) and departure (egress) measurements, we use a phase plot representation [14]. To construct the phase plot, we consider the time delay between consecutive probe packets as they arrive at the router (ingress spacing $= s_i$) as the x dimension on the plot and consider the spacing of the same packets as they exit the router (egress spacing $= s_e$) as the y dimension. From these measurements, there are three possibilities for the ratio $s_r = s_e/s_i$. If $s_r = 1$ then spacing remained unchanged by the router. If $s_r > 1$ then other packets enqueued between the two packets causing *expansion*. If $s_r < 1$ then the first packet was delayed because of a queue

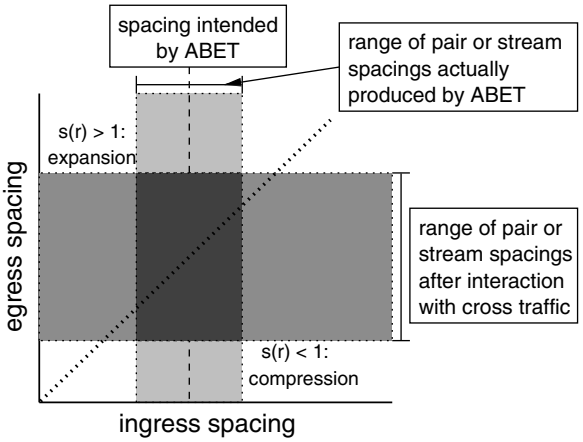

Fig. 1. Application of phase plots to available bandwidth estimation tool analysis and calibration.

that has diminished by the time the second packet arrives, causing *compression*.

Fig. 1 depicts how we use phase plots for ABET calibration. The ingress dimension of the plot should reveal any differences between spacings that are intended by the ABET, and the spacings actually produced. This provides the ability to assess bias introduced into the measurement process by imprecise commodity hardware and operating systems. The egress dimension of the plot shows the spacings on which inferences are made by the receiver after interaction with cross traffic, though they may differ from the spacings actually measured by the receiver. Note that while some ABETs do not make inferences directly from these spacings (e.g., PATHLOAD), they play a key role in what an ABET infers. Therefore, these spacings enable calibration of both the inference method as well as providing a baseline for calibrating the receiving host.

4. Calibration experiments

The objective of our calibration study of PATHLOAD and SPRUCE was to examine the tools under a number of increasingly more complex traffic and path conditions and to understand where they work well and where they work poorly, and why. In each experiment we evaluate the tool's ability to report AB within a range of 10% of the tight link capacity. This threshold is chosen as an arbitrary reference point in the sense that a threshold would typically be chosen based on specific requirements of a target application. We required that estimates be consistently within this window of accuracy for a series of estimates reported by an ABET over the duration of an experiment. Without the property of *consistent accuracy*, ABETs are unlikely to be used in applications such as in re-optimization of an overlay network.

4.1. Testbed setup

Our primary testbed configuration consisted of variations on a dumbbell-like topology with an OC-3 narrow link as depicted in Fig. 2. We used a total of six setups, including three traffic scenarios: constant bit-rate (CBR) traffic of 50 Mb/s (UDP traffic with uniformly spaced 1500 byte packets), 19 long-lived TCP flows in a single direction, 19 long-lived TCP flows in each direction, and three variants of a setup using web-like traffic with file sizes drawn from a heavy-tailed distribution to produce self-similar traffic. In all cases, the direction of interest was left to right in the figure (i.e., CBR, single direction long-lived TCP connections, and web-like traffic traveled left to right). Cross traffic was generated by hosts running Harpoon [25] (web-like traffic) or Iperf [26] (infinite source and constant bit-rate traffic).[3] We used an Adtech SX-14 hardware propagation delay emulator configured to add a delay of 10 ms in each direction for all experiments.

To create increasingly more complex path conditions, we considered the following three topological setups.

4.1.1. Topology 1 (narrow and tight link are the same, homogeneous round-trip time)

Probe traffic was configured to cross the Gigabit Ethernet link directly connecting routers at hops A and C. No cross traffic was routed across this link. CBR and long-lived TCP connection traffic crossed the Cisco 12000 at hop B, while web traffic was configured to use the two Cisco 7200's and the Cisco 12000 at hop B, but not the direct link to hop C. Our decision to route probe traffic direction from hop A to hop C caused the tight link and narrow link to be identical in the CBR, long-lived TCP source, and basic web-like traffic scenarios. When using web-like cross traffic in this setup, we configured Harpoon to produce an average of 50 Mb/s.

4.1.2. Topology 2 (narrow and tight link are not the same, homogeneous round-trip time)

Using web-like cross traffic, we routed probe traffic across a Fast Ethernet link between hops A and B, but configured cross traffic not to use this link. In this experiment, we also configured the cross traffic sources to produce approximately 100 Mb/s of traffic on the OC-3 link between hops D and E, causing the Fast Ethernet link to be the narrow link, but the OC-3 to be the tight link.[4]

4.1.3. Topology 3 (narrow and tight link are not the same, heterogeneous round-trip time)

Using again web-like cross traffic, we configured our Linux traffic generation hosts with NetPath [27] to emulate round-trip times of 20, 50, 80 and 110 ms. We also attached

[3] Our traffic generator hosts were identically configured workstations running either Linux 2.4 and FreeBSD 5.3. The workstations had 2 GHz Intel Pentium 4 processors, 2 GB of RAM and Intel Pro/1000 cards (with interrupt coalescence disabled). Each system was dual-homed, so that all management traffic was on a network separate from the one depicted in Fig. 2. Probe traffic systems were identical to the traffic generators and ran FreeBSD 5.3.

[4] We verified in each experiment that, over each tool measurement interval, the tight link was always the OC-3 link.

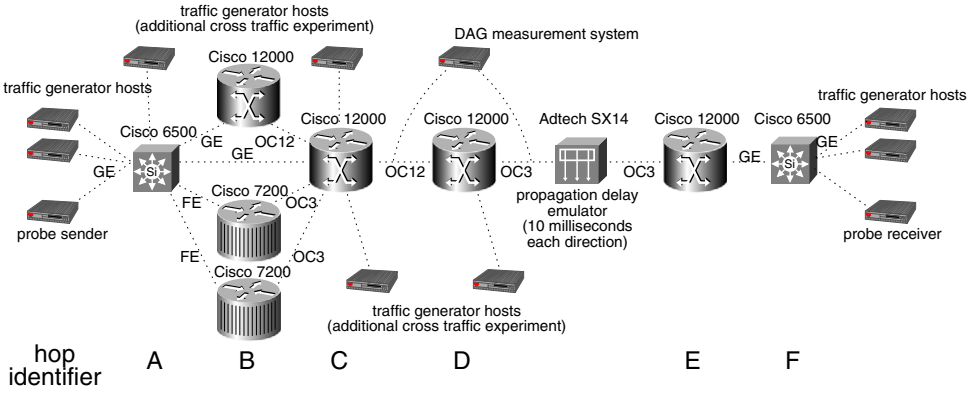

Fig. 2. Experimental testbed. Cross traffic scenarios consisted of constant bit-rate traffic, long-lived TCP flows, and bursty web-like traffic. Cross traffic flowed across one of three routers at hop B, while probe traffic normally flowed directly between hops A and C over a Gigabit Ethernet (GE) link. In a setup forcing the narrow and tight links to be distinct physical links, probe traffic crosses a Fast Ethernet (FE) link between hops A and B. In a setup considering additional web-like cross traffic, hosts shown attached at hops A, C, and D generate traffic that persists on shared links for one hop. Optical splitters connected Endace DAG 3.5 or 3.8 measurement cards to the testbed between hops C and D, and hops D and E. The bulk of cross traffic and probe traffic flowed left to right.

additional hosts at hops A, C, and D to generate cross traffic that traveled across all links between hops A and C (sharing the link with probe traffic) or the OC-12 link between hops C and D.

Critical to our calibration methodology was the ability to take high accuracy measurements in our test environment. To do this we attached optical splitters and En-dace DAG 3.5 packet capture cards (affording timestamping accuracy on the order of single microseconds [28][5]) to monitor the links between hops C and D, and hops D and E. We used these monitoring points to create phase plots and measure utilization on the tight OC-3 link. This configuration gave us ground truth measurements well beyond the coarse-grained SNMP measurements used in prior *in situ* studies of ABETs.

4.2. ABET calibration: Comparison

The calibration framework described in Section 3 directs our evaluation process. We begin by assessing the capabilities of the end hosts running the ABETs. Sources of potential bias introduced by end hosts include OS context switches and other system capability/OS effects such as network adapter interrupt coalescence [7,29,30]. Our interest is not in untangling the details of each source of host system bias, rather it is in understanding the overall impact.

In our experiments below, we considered topology 1 and collected traces from a single PATHLOAD fleet (1200 probe packets of 1309 bytes), and a series of 12 SPRUCE runs (1200 packet pairs, each packet of length 1500 bytes) with constant bit rate cross traffic of 50 Mb/s flowing across the narrow link during each test. If the host systems emitted packets without bias, we would expect ingress spacings for both tools to be tightly clustered around the intended value of 80 μs.

The phase plots for these experiments shown in Fig. 3 immediately expose two potential sources of measurement bias. First, it is easy to see that for each ABET there is a wide range of interpacket spacings on ingress which can be attributed to the sending host. Second, it is also evident that an effect of the CBR cross traffic is to cause a respacing of probe packets on egress to either back-to-back (70 μs for PATHLOAD packets, 80 μs for SPRUCE packets) or with one cross traffic packet interposed (150 μs for PATHLOAD, 160 μs for SPRUCE). Closer examination reveals that packets spaced farther apart by the ABET are more likely to experience expansion by a cross traffic packet than to be transmitted back-to-back on the tight link. This can be seen in Fig. 3a by the perceptible shift to the right in the upper cluster of points. A similar shift exists in Fig. 3b. Finally, we note that points below the diagonal line in Fig. 3a represent evidence for compression in PATHLOAD streams. We quantify the prevalence of this effect below.

To further explore the problem of bias imposed by probe senders, we collected several thousand packet spacing measurements from SPRUCE and PATHLOAD and compared each spacing with the spacing measured at the DAG monitor between hops C and D. Fig. 4a shows a representative histogram of differences between the spacing measured using gettimeofday () at PATHLOAD.[6] From these measurements, we conclude that while the magnitude of individual errors can be quite significant, the mean deviation is close to zero.

Next, we examined the measurements at the receiving application. PATHLOAD timestamps outgoing/incoming packets using the `gettimeofday ()` operating system

[5] As a consistency and calibration check, we also captured traces using Endace DAG 3.8 cards, which employ a higher frequency clock, and have somewhat different architectural features than the DAG 3.5. The resulting phase plots were consistent with those produced using the DAG 3.5. Experiments described below employ the DAG 3.5 cards unless otherwise specified.

[6] We modified SPRUCE and PATHLOAD to log these timestamps.

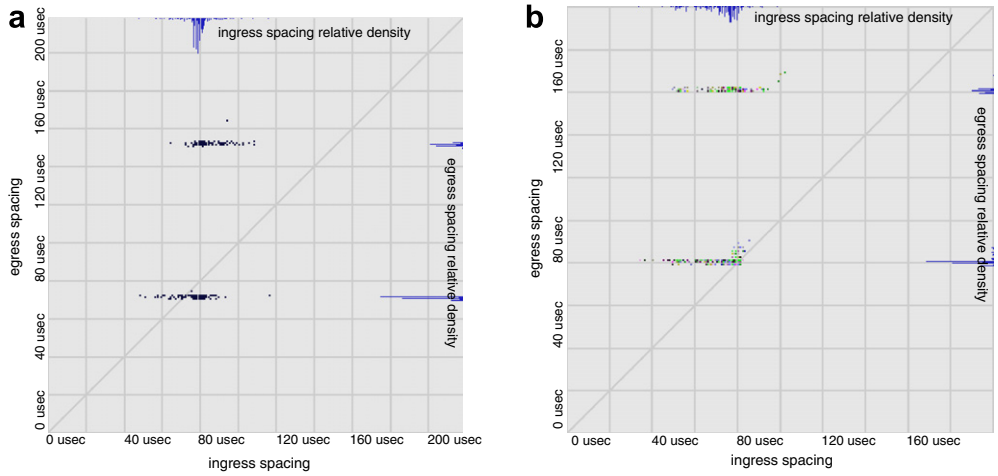

Fig. 3. Phase plots of PATHLOAD and SPRUCE streams. Grid lines are separated by 20 μs for each plot. CBR cross traffic of 50 Mb/s, with uniform UDP packets 1500 bytes (not shown in plots) causes bimodal output spacing distribution of probe traffic. Target input spacing for each tool is 80 μs. Note the slightly different scale for each plot. (a) Phase plot produced from one PATHLOAD fleet (1200 probe packets of length 1309 bytes). (b) Phase plot produced from 12 SPRUCE runs (1200 packets pairs, packets of length 1500 bytes).

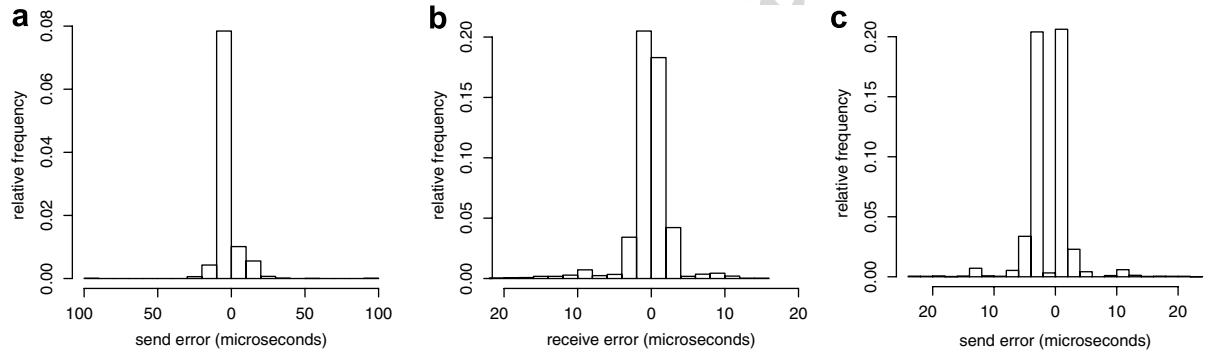

Fig. 4. Relative frequencies of errors between send or receive packet spacings and spacings measured at DAG monitor. (a) Distribution of errors at FreeBSD 5.3 sender with ntel Pro/1000 Gigabit Ethernet adapter and DAG 3.5 between hops C and D. (b) Distribution of errors at FreeBSD 5.3 receiver with Intel Pro/1000 Gigabit Ethernet adapter and DAG 3.5 between hops D and E. (c) Distribution of errors at Linux 2.4 sender with Intel Pro/1000 Gigabit Ethernet adapter and DAG 4 attached directly to network interface.

call. SPRUCE timestamps outgoing packets using the `gettimeofday ()` system call and incoming packets receive timestamps in the OS interrupt handler. Timestamps used for both these tools are of microsecond precision (though not necessarily microsecond accuracy). Comparing timestamps measured upon packet receive with timestamps measured at the DAG monitor between hops D and E (i.e., the egress spacings of Fig. 3 compared with application-measured receive spacings), we obtain a result similar to the sender. Fig. 4b shows a representative histogram of differences in packet spacings measured at the probe receiver versus the same spacings measured at the DAG monitor. The magnitude of error is smaller than that on the sender and the mean deviation is close to zero.

As a final calibration check, and to test whether these results were unique to the hardware and OS configuration used, we attached a DAG 4 (Gigabit Ethernet) monitor directly to the Intel Pro/1000 on a Linux 2.4 workstation and collected additional measurements using SPRUCE. A histogram of differences between spacings measured at SPRUCE and spacings measured at the DAG 4 is shown in Fig. 4c. Again, the mean deviation is close to zero. Packet receive errors on the Linux 2.4 system (not shown) are also close to zero mean. Table 1 summarizes these results.

Even though the averaged behavior of probe streams tends toward the intended value, bias on individual probes can still have a significant detrimental effect on the operation of PATHLOAD and SPRUCE. Since the source code for each tool treats each timestamp as accurate, while utilizing the microsecond precision supplied by the gettimeofday () system call, even very tiny differences may lead to poor AB estimation. Also, since errors are widely distributed around zero and may not be uniformly distributed, simple filtering schemes may be ineffective.

4.3. ABET calibration: Algorithmic adjustment

Although phase plots are not new to the networking community, as illustrated above, their use in ABET calibration has been very beneficial. They not only helped us

Table 1
Summary of errors between packet spacings measured at application send and receive, and DAG monitors

	DAG 3.5/3.8 (OC-3/12) FreeBSD 5.3		DAG 4 (Gigabit Ethernet) Linux 2.4
	Send error	Receive error	Send error
Minimum	−93.00	−20.00	−24.00
Median	−2.00	0.00	−2.00
Mean	−1.54	0.15	−0.61
Maximum	100.00	18.00	23.00

All values are in microseconds. Negative values indicate that a larger spacing was measured at DAG monitor than in the application.

to expose end-host limitations of generating precise streams and to identify the resulting bias, but by studying the egress spacings, we were also able to gain an expectation of what the receiving host *should have* measured and realized that considering compression events is important. In summary, phase plot analysis resulted in the following observations:

- The error introduced by end hosts has approximately zero mean when multiple measurements are taken.
- The relationship between input and output probe rates and available bandwidth described in Eq. (4) invites refinements.
- Both compression and expansion are indicative of congestion along a measured path.

These observations lead us to propose a calibrated algorithm for measuring available bandwidth. We first test how quickly the mean deviation of measurements converges, on average, to zero. That is, how many packets should comprise a stream, at minimum, in order for the error to be less than some threshold? To answer this question, we created a tool to send packet streams of length 100 packets at four target spacings of 60, 80, 100, and 120 μs, in separate experiments. We ran the tool under topology 1 with no cross traffic to collect approximately 1000 packet stream measurements (i.e., about 100,000 packets per experiment).[7] For each packet stream, we counted the number of packets required for the mean deviation between spacings measured at the DAG monitor and timestamps generated by the application to be less than 1 μs. Fig. 5 plots the cumulative distribution of stream lengths required for each target spacing. The distributions show that the mean error converges to zero quite quickly and that packet streams of at least length 20 packets appear to be sufficient.[8] However,

[7] Phase plots from these experiments are similar to those shown in Fig. 3.

[8] Although we do not show detailed results here, we also ran experiments with interrupt coalescence (IC) enabled on packet transmit, with an absolute timer of 64 μs (the default value). The absolute timer sets the maximum amount of time that a packet will be held awaiting another packet to send before triggering an interrupt in the device driver.The results for this configuration show that target spacings that are approximately 20–40 μs above the IC absolute timer cannot be met, and no packet stream of reasonable length can reduce the error to an acceptable range. We therefore ran all experiments with IC disabled.

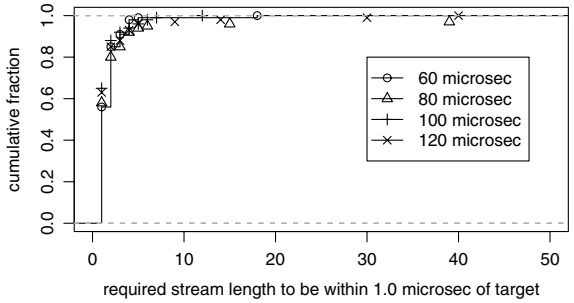

Fig. 5. Cumulative distribution of stream lengths required to cause mean sending error to be within 1 μs of target. Target spacings are 60, 80, 100, and 120 μs.

there remain tradeoffs for ABET methodologies using packet streams. While shorter streams may reduce the intrusiveness of the tool, and may reduce measurement latency, the averaging time scale is also reduced, theoretically resulting in greater measurement variance (Eq. (1)).

At the base of our proposed algorithm is a version of Eq. (4), modified to consider absolute difference in average input and output spacings. The absolute difference is used because both compression and expansion (or the combination of both in a given measurement period) must be considered as an indication of congestion. Average spacings are used since we have shown that individual spacings are subject to significant error. In the formulation below, we consider the absolute difference between the average input and output spacings as an indication of whether the input rate was above the available bandwidth along the path (analogous to Eq. (4)):

$$|\overline{g_{\text{in}}} - \overline{g_{\text{out}}}| = \begin{cases} \leqslant \zeta * \overline{g_{\text{in}}} & r_{\text{in}} \leqslant A \\ > \zeta * \overline{g_{\text{in}}} & r_{\text{in}} > A. \end{cases} \quad (7)$$

The value ζ is a threshold parameter used to determine how far apart the average send and receive spacings can be while still considering the input stream to be below the level of spare capacity along the measurement path.

Like PATHLOAD, our algorithm for finding the available bandwidth is iterative. First, we set the target send spacing, g_{target}, to be some minimum value (effectively setting the maximum AB measurable), then proceed as follows.

(1) Send probe stream, measuring $\overline{g_{\text{in}}}$ and $\overline{g_{\text{out}}}$ at send and receive hosts, respectively.
(2) If the absolute difference in average input and output spacings is above the ζ threshold of the input spacing Eq. (7), increase g_{target} by $\frac{|\overline{g_{\text{in}}} - \overline{g_{\text{out}}}|}{2}$, wait a configurable amount of time, and go to previous step.
(3) Otherwise, update an exponentially weighted moving average (EWMA) (with parameter α) with the estimate r_{in}. Report the updated EWMA as the estimate of AB.

The reason for using an exponentially weighted moving average on the individual estimates is that there will still be

outliers in measurements. Even though Fig. 5 implies that a stream length of 20 packets is generally sufficient, due to the unpredictability of commodity systems, care is still required.

We consider the above algorithm to be a "calibrated Pathload" and have implemented it in a tool called YAZ.[9] The source code for YAZ is available to the research community for evaluation at http://wail.cs.wisc.edu.

4.4. Experimental evaluation

We compared the accuracy of PATHLOAD, SPRUCE, and YAZ using the different scenarios described in Section 4.1. For the CBR and long-lived TCP source experiments, we continuously collected AB estimates from each tool for 10 min, discarding the first 30 s and last 30 s. For the web traffic setups, we continuously collected AB estimates from each tool for 30 min, also discarding the first 30 s and last 30 s. For the comparisons below, we compute the actual available bandwidth using the DAG monitor between hops D and E for the exact interval over which a tool produces an estimate.[10] For each experiment, we consider the fraction of estimates that fall within a range of 10% of the tight link capacity. Since our tight link is OC-3 (149.76 Mb/s before Cisco HDLC overhead), this window is ≈15 Mb/s.

For all experiments, YAZ was configured with $\alpha = 0.3$ in its exponentially weighted moving average and the threshold parameter ζ was set to be equivalent to a rate of 1 Mb/s, which we found to be a robust setting over our topologies and traffic scenarios. α of 0.3 produced minimum MSE over the collection of experiments. We set YAZ's stream length to 50 packets. For SPRUCE, we use 149.76 Mb/s as the tight link capacity in Eq. (3) for all experiments except for the second web-like traffic scenario, in which we set it to 97.5 Mb/s (the narrow link is Fast Ethernet) and use the default value of 100 samples to compute an estimate of AB. For PATHLOAD, we used default parameters, and in the initial comparison with YAZ, we set the stream length to 50 packets, while leaving the number of streams per fleet at the default value of 12. We report the midpoint of PATHLOAD's estimation range as the AB estimate which, as we discuss below, is generally favorable to PATHLOAD.

Results for all the experiments are shown in Fig. 6. The results for constant bitrate traffic in topology 1 (Fig. 6a) show that both YAZ and PATHLOAD perform with similar accuracy, coming quite close to the true AB. However, fewer than 60% of SPRUCE estimates are within the 10% acceptance range.

The two long-lived TCP traffic scenarios in topology 1, in some ways, create the most pathological cross traffic conditions due to the frequent traffic oscillations on the tight link. Fig. 6b plots results for the setup with TCP flows in a single direction. The YAZ estimates are fully within the 10% threshold, while more than 90% of PATHLOAD's estimates are within this bound. Only about 20% of SPRUCE estimates fall within the acceptable range. For the bi-directional long-lived TCP flows, YAZ and PATHLOAD perform similarly, with approximately 90% of estimates falling within the 10% acceptance range. Again, very few estimates produced by SPRUCE fall within the 10% range.

For the web-like cross traffic in topology 1 experiment (Fig. 6c), approximately 75% of estimates produced by YAZ are within the acceptance range compared to about 50% of PATHLOAD estimates and about 40% of SPRUCE estimates. We also ran PATHLOAD in this setup again, setting the stream length to be 100 packets (the default in the PATHLOAD source code). Fig. 6e shows the result of this experiment, comparing the YAZ and SPRUCE results from Fig. 6d. We see that the accuracy of PATHLOAD improves by about 15%.

The results for the case of web-like cross traffic in topology 2 are shown in Fig. 6f. In this setup, PATHLOAD underperforms both YAZ and SPRUCE, with about 65% of YAZ estimates and about 55% of SPRUCE estimates falling within the 10% threshold, but only about 40% of PATHLOAD estimates falling within this range. A closer look at the PATHLOAD results revealed that it took longer on average to converge on an estimation range, and convergence times were more variable than in any other setup. Since AB is a moving target, these increased convergence times led to poor estimates. Finally, Fig. 6g shows results for the web-like cross traffic in topology 3. In this setup, about 80% of YAZ estimates are within the acceptance range, compared with about 50% for PATHLOAD and 40% for SPRUCE.

4.5. Detailed analysis of experimental results

4.5.1. Reported range of PATHLOAD

Although we do not report detailed results, we also examined how often the actual AB fell in the range reported by PATHLOAD. For the constant bitrate and long-lived TCP experiments, the actual value is rarely within PATHLOAD's range. The reason is that its range was often a single point, but not equal to the actual AB. For the three self-similar web-like traffic scenarios, the actual AB is, at best, within PATHLOAD's range 58% of the time (53/92 estimates, for topology 3). For these experiments, the width of the range varies greatly, preventing general explanation. In the end, our focus on comparing the midpoint of PATHLOAD's estimation range with the actual AB is favorable to PATHLOAD. We plan to more carefully evaluate AB variation as future work.

[9] Although the name YAZ is reminiscent of tool names starting with "yet another ...", our tool is actually named after the baseball great, Carl Yastrzemski.

[10] We include Cisco HDLC overheads (9 bytes per packet) in this computation. Since we control packet payloads, we limit any hidden effects due to SONET character stuffing.

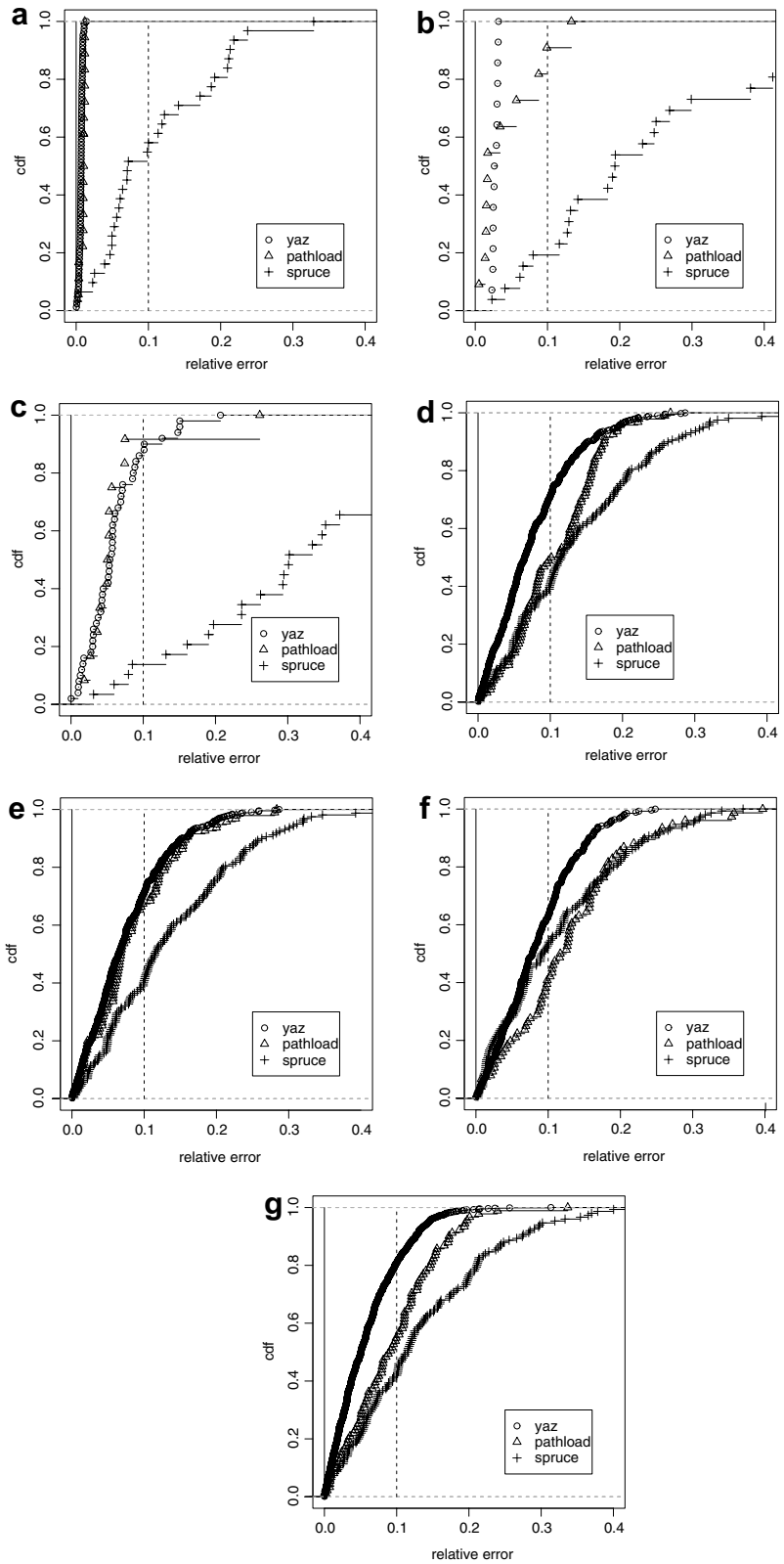

Fig. 6. Comparison of available bandwidth estimation accuracy between YAZ, PATHLOAD, and SPRUCE for the constant bit rate and long-lived TCP traffic scenarios. True available bandwidth is computed using DAG traces over the same interval on which a tool estimation is performed. Dashed vertical line at $x = 0.1$ indicates 10% desired accuracy threshold. (a) Constant bit rate cross traffic of 50 Mb/s (Topology 1). (b) Long-lived TCP sources in one direction (left to right in Fig. 2) (Topology 1). (c) Long-lived TCP sources in two directions (Topology 1). (d) Web-like cross traffic produced by Harpoon with average rate of 50 Mb/s (Topology 1). (e) Comparison of YAZ, PATHLOAD and SPRUCE for web-like traffic when PATHLOAD is configured for streams of length 100 packets (Topology 1). (YAZ and SPRUCE curves are same as in d). (f) Web-like traffic with narrow link (Fast Ethernet) and tight link (OC-3) as distinct physical links (Topology 2). (g) Web-like traffic with additional points of cross traffic and a diversity of round trip times (Topology 3).

4.5.2. The effect of PATHLOAD's stream length

To better understand the underlying reason for PATHLOAD's improved AB estimates with a larger number of probes, we examined a series of streams emitted by PATHLOAD in the web-like traffic scenarios. We calculated the PCT and PDT metrics for each stream using the DAG monitor between hops C and D – *before* any interaction with cross traffic. Table 2 summarizes these results. The mean PCT for stream lengths of 50 is close to the threshold of 0.55 that considers the stream to reflect an increasing trend in OWD. With the longer stream length of 100, the mean PCT is further away from the threshold. This shift suggests that the longer stream has an important side-effect on the PCT metric, and that there is substantial bias introduced at the sending host. From the results in Table 2, stream length appears to have less of an impact on the initial bias of the PDT metric. For each stream length, at least 15% of the streams departed the sending host with at least one of the metrics exceeding its threshold.

4.5.3. PATHLOAD stream compression

Table 3 quantifies the prevalence of compression in PATHLOAD streams. We compared the spacing intended by PATHLOAD for each stream with the spacings measured at the DAG monitor between hops C and D. The values for each traffic scenario in the table show the fraction of streams for which there was an overall effect of compression. We see that, in general, about 20% of all streams are compressed. In the case of long-lived TCP flows in one direction, the queue at the tight link is usually increasing as the flows increase their windows until loss occurs. The fact that a non-negligible fraction of streams over each scenario experience compression supports Eq. (7) as a key distinguishing feature between YAZ and PATHLOAD. It also, in part, explains why YAZ outperforms PATHLOAD.

4.5.4. Tool accuracy without cross traffic

For PATHLOAD, using the median OWD over a window of samples appears to be a critical component of at least the PCT formulation. A technique like SPRUCE, on the other hand, is not insulated from individual spacing and measurement errors; even a difference of a single microsecond (assuming a target spacing of 80 μs) can lead to an estimation error of nearly 2 Mb/s. For example, we ran SPRUCE in the testbed while not introducing any cross traffic. In this setup, the tool should invariably report close to 149.76 Mb/s. Over 10 consecutive runs, SPRUCE produced estimates ranging from 134.4 to 153.8 Mb/s, with a mean of 149.4 Mb/s and a standard deviation of 5.42. These inaccuracies are entirely due to measurement error. While most estimates are close to the true AB, the worst estimate is just beyond the desired 10% accuracy range. (For similar zero cross traffic experiments using PATHLOAD and YAZ, the estimates were consistently within 1–2% of the true AB.)

4.5.5. The asymptotic nature of SPRUCE

Lastly, as we noted in Section 2, the SPRUCE formula is an asymptotic result. During both long-lived TCP source experiments, SPRUCE reported negative estimates. Even with perfect measurement devices, it is not clear, *a priori*, how long a tool like SPRUCE should collect samples. While the default value of 100 may be sufficient for certain traffic scenarios, it is clearly insufficient for others.

4.5.6. Estimation latency and overhead

Lastly, we compare estimation latency, the average number of probes emitted per estimate, and the number of estimates produced during the first web-like traffic scenario. Table 4 summarizes these results, which are qualitatively similar for other traffic scenarios. We see that YAZ produces estimates more quickly, thus producing many more estimates over the duration of the experiment. PATHLOAD and YAZ operate in an iterative fashion, and we see from the table that YAZ, on average, requires fewer cycles to arrive at an estimate.

Table 2
Mean and standard deviation of PCT and PDT values for streams of length 50 or 100 packets upon departure (prior to interaction with cross traffic) for web-like traffic scenarios

Stream length	PCT		PDT	
	μ	σ	μ	σ
50	0.4	0.14	0.04	0.28
100	0.26	0.14	−0.07	0.27

Table 3
Prevalence of compression in PATHLOAD streams for all six cross traffic scenarios

Traffic scenario	Fraction of compressed streams
CBR traffic of 50 Mb/s (Topology 1)	0.212
Long-lived TCP, one direction (Topology 1)	0.077
Long-lived TCP, two directions (Topology 1)	0.260
Web-like traffic (Topology 1)	0.233
Web-like traffic (Topology 2)	0.220
Web-like traffic (Topology 3)	0.219

Table 4
Comparison of number of estimates produced, latency, number of packets emitted per iteration (PATHLOAD and YAZ), and average number of packets emitted per estimate for each ABET for web-like traffic in topology 1

	Estimates produced	Latency $\mu(\sigma)$ (s)	Iterations per estimate $\mu(\sigma)$	Mean packets per estimate
PATHLOAD ($K = 100$)	96	17.7 (3.8)	8.4 (4.8)	10,080
PATHLOAD ($K = 50$)	97	17.6 (3.8)	8.8 (4.2)	5280
SPRUCE	156	10.9 (0.9)	NA	200
YAZ	446	3.8 (1.5)	6.1 (8.8)	366

Considering tool parameters and the mean number of iterations, we arrive at the mean number of packets required for each estimate. For much higher accuracy, YAZ uses packets roughly of the same order of magnitude as SPRUCE, but at least an order of magnitude fewer packets than PATHLOAD. If PATHLOAD and SPRUCE represent a tradeoff between measurement accuracy and overhead, our results for YAZ suggest that this tradeoff is not fundamental. For a SPRUCE-like budget, YAZ is more accurate than PATHLOAD, sometimes significantly so.

4.6. Limitations of YAZ

There are some limitations to YAZ that we have yet to fully examine. First, we do not yet have a complete understanding of how to set the ζ threshold parameter and its sensitivity to particular operating system and hardware characteristics and cross traffic conditions. Second, since we use the mean spacing measured at sender and receiver, we cannot detect intra-stream indications of congestion that may be "washed out" over the duration of the stream. We can only detect either persistent expansion or compression of a probe stream. Given our current understanding of the nature of the errors introduced by commodity end hosts, we may not be able to do significantly better. Third, the initial minimum value of g_{in} is specified by the user. Determining how best to automatically set this parameter for a range of environments is an area for future work. Finally, our calibration study has focused on average AB over a time interval. PATHLOAD reports a variation range for AB, and Jain and Dovrolis [17,31] have shown that the variation of AB is an important measurement target. Extending our calibration study to consider AB variation is a subject for future work.

5. Summary and conclusions

The primary objective of this paper is to highlight calibration as a key component in the design, development and rigorous testing of available bandwidth measurement tools. We advocate the use of controlled laboratory experiments as a means for partially overcoming the limitations that are inherent in standard ns-type simulations. While *in vitro*-like testing is unlikely to fully replace experiments *in situ*, it offers complete control, full instrumentation and repeatability which are all critical to tool calibration. We note that the laboratory setups used in our study can be recreated by other researchers [15].

We propose a framework for the calibration of ABETs. Our case study exposes potential biases and inaccuracies in ABE due to the use of commodity systems for high fidelity measurement and/or inaccurate assumptions about network system behavior and traffic dynamics. As a result of these observations, we developed a calibrated Pathload-like tool called YAZ, which is consistently more accurate than prior ABETs. For example, in a 30 min-long experiment in a traffic scenario using Internet-like bursty cross traffic, 81% of YAZ AB estimates are within 10% of the true value, while only 57% of PATHLOAD estimates and 41% of SPRUCE estimates fall within the same window of accuracy. For this significantly higher level of accuracy, YAZ uses packets roughly of the same order of magnitude as SPRUCE, but at least an order of magnitude fewer packets than PATHLOAD. If PATHLOAD and SPRUCE represent a tradeoff between measurement accuracy and overhead, our results for YAZ suggest that this tradeoff is not fundamental. We believe that YAZ is representative of the type of active measurement tool that can be expected as a result of insisting on more stringent calibration.

We also advocate the use of phase plots to analyze and visualize the fine-grained measurements resulting from our ABET experiments. We show how phase plots were instrumental in exposing existing ABET bias and errors, and the qualitative insight rendered by them was key to the resulting design of YAZ. We do not claim that such plots are a panacea, exposing all sources of bias and error for all active measurement tools. However, we believe that there is an outstanding need for new flexible analysis and visualization tools capable of more fully exposing the enormous quantity of high fidelity measurement data that can be collected in the calibration framework we advocate in this paper. While the focus is on a calibration strategy tuned to the problem of ABE in this paper, we intend to generalize our approach to additional active measurement-based tools that attempt to infer network internal characteristics.

References

[1] V. Paxson, Strategies for sound Internet measurement, in: Proceedings of ACMSIGCOMM Internet Measurement Conference '04, 2004.
[2] S. Floyd, V. Paxson, Difficulties in simulating the Internet, in: IEEE/ACM Transactions on Networking, vol. 9, No. 4.
[3] S. Floyd, E. Kohler, Internet research needs better models, in: Hotnets-I, Princeton, NJ, 2002.
[4] A. Akella, S. Seshan, A. Shaikh, An empirical evaluation of wide-area internet bottlenecks, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '03, 2003.
[5] R. Carter, M. Crovella, Measuring bottleneck link speed in packet-switched networks, in: Proceedings of Performance '96, Lausanne, Switzerland, 1996.
[6] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, in: IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling 21 (6).
[7] M. Jain, C. Dovrolis, End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput, in: Proceedings of ACM SIGCOMM '02, Pittsburgh, PA, 2002.
[8] K. Lakshiminarayanan, V. Padmanabhan, J. Padhye, Bandwidth estimation in broadband access networks, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '04, Taormina, Sicily, Italy, 2004.
[9] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidth bottlenecks, in: Proceedings of Global Internet Symposium, 2000.
[10] V. Paxson, End-to-end Internet packet dynamics, in: Proceedings of ACM SIGCOMM '97, Cannes, France, 1997.
[11] V. Riberio, R. Riedi, R. Baraniuk, J. Navratil, L. Cottrell, pathChirp: Efficient available width estimation for network paths, in: Proceedings of Passive and Active measurement Workshop, 2003.

[12] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, K.C. Claffy, Comparison of public end-to-end bandwidth estimation tools on high-speed links, in: Proceedings of Passive and Active Measurement Workshop '05, 2005.
[13] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '03, Miami, FL, 2003.
[14] J. Sommers, P. Barford, W. Willinger, SPLAT: A visualization tool for mining Internet measurements, in: Proceedings of Passive and Active Measurement Conference, '06, 2006.
[15] The Wisconsin Advanced Internet Laboratory. Available from: http://wail.cs.wisc.edu, 2006.
[16] X. Liu, K. Ravindran, B. Liu, D. Loguinov, Single-hop probing asymptotics in available bandwidth estimation: sample-path analysis, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '04, Taormina, Sicily, Italy, 2004.
[17] M. Jain, C. Dovrolis, Ten fallacies and pitfalls on end-to-end available bandwidth estimation, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '04, Taormina, Sicily, Italy, 2004.
[18] S. McCanne, S. Floyd, UCB/LBNL/VINT Network Simulator – ns (version 2). Available from: http://www.isi.edu/nsnam/ns/.
[19] PLANETLAB – an open platform for developing, deploying, and accessing planetary-scale services. Available from: http://www.planet-lab.org, 2006.
[20] DETER: A laboratory for security research. Available from: http://www.isi.edu/deter/, 2006.
[21] Emulab — network emulation testbed. Available from: http://www.emulab.net, 2006.
[22] L. Le, J. Aikat, K. Jeffay, F. Smith, The effects of active queue management on web performance, in: Proceedings of ACM SIGCOMM '03, Karlsruhe, Germany, 2003.
[23] Y.-C. Cheng, U. Hölzle, N. Cardwell, S. Savage, G. Voelker, Monkey see, monkey do: A tool for TCP tracing and replaying, in: Proceedings of the USENIX 2004 Conference, 2004.
[24] F. Hernandez-Campos, F. Smith, K. Jeffay, How real can synthetic network traffic be?, in: Proceedings of ACM SIGCOMM '04 (Poster Session), 2004.
[25] J. Sommers, P. Barford, Self-configuring network traffic generation, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '04, 2004.
[26] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, K. Gibbs, Iperf 1.7.0 – the TCP/UDP bandwidth measurement tool. Available from: http://dast.nlanr.net/Projects/Iperf, 2006.
[27] S. Agarwal, J. Sommers, P. Barford, Scalable network path emulation, in: Proceedings of IEEE MASCOTS '05, 2005.
[28] S. Donnelly, High precision timing in passive measurements of data networks, Ph.D. thesis, University of Waikato, 2002.
[29] G. Jin, B. Tierney, System capability effects on algorithms for network bandwidth measurement, in: Proceedings of ACM SIGCOMM Internet Measurement Conference '03, 2003.
[30] R. Prasad, M. Jain, C. Dovrolis, Effects of interrupt coalescence on network measurements, in: Proceedings of Passive and Active Measurement Workshop '04, 2004.
[31] M. Jain, C. Dovrolis, End-to-end estimation of the available bandwidth variation range, in: Proceedings of ACM SIGMETRICS '05, Banff, Alta., Canada, 2005.