A Proposed Framework for Calibration of Available Bandwidth Estimation Tools

Joel Sommers Paul Barford
University of Wisconsin-Madison
jsommers,pb@cs.wisc.edu

Walter Willinger
AT&T Labs-Research
walter@research.att.com

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 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.*

1. Introduction

Calibration strategies for Internet measurement tools are essential for detecting inaccuracy in the underlying data, and misconceptions or errors in their analysis [20]. 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 in [20], 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 parts of the live Internet. Ns-type simulations are attractive since they have the advantage of simplified implementations and complete experimental control. However, by definition they are an abstraction of networking reality [10] 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 parts of 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 parts 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 [6, 7, 12, 13, 15, 18, 19, 22, 23, 27]. 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.

Using an openly available laboratory testbed [4], we apply our calibration strategy through a series of experiments to two ABETs, SPRUCE [27] and PATHLOAD [13], 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 using multiple tools and show 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 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. Full details and results of our study are found in [25].

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 [6, 7, 12, 13, 15, 18, 19, 22, 23, 27]), two recent tools, PATHLOAD [13] and SPRUCE [27], 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 [23]. 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.

End-to-end AB is informally defined as the minimum spare capacity on an end-to-end path over a given time interval. The link with smallest AB is referred to as the *tight link*, while the link with minimum capacity along a path is referred to as the *narrow link*. These definitions avoid the ambiguous term *bottleneck link* [13]. They also help to avoid any implicit assumption that the tight link is necessarily the narrow link. Existing tools for measuring AB 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.

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

[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 [27].

measured at the receiver, the AB is calculated as:

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

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

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 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:

$$\frac{r_{in}}{r_{out}} = \begin{cases} \leq 1 & r_{in} \leq A \\ > 1 & r_{in} > A \end{cases} \quad (2)$$

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. Details of the metrics that PATHLOAD uses to infer OWD trends are found in [13].

3. Calibration Framework

Comparison with a standard and subsequent adjustment of an ABET's algorithm or parameters are complementary activities. 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.*, 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 timestamp accuracy (and in some cases, synchronization) is sufficient.

2. Assumptions about and/or abstract models for the behavior of routers and switches 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 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. The second two issues 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, as far as possible, realistic 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 [17] or *in situ* settings such as PlanetLab [3], 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 this first 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 tech-

niques 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 [1, 2, 4]. Although similar environments have been used successfully in recent studies [16, 23], 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 [8, 11, 24] 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 scatter or phase plot representation [26]. 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 that has diminished by the time the second packet arrives, causing *compression*.

Figure 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

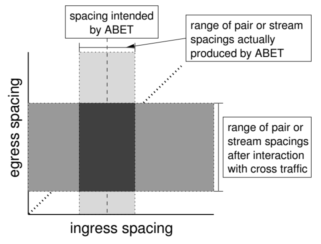

spacing intended by ABET
range of pair or stream spacings actually produced by ABET
egress spacing
range of pair or stream spacings after interaction with cross traffic
ingress spacing

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

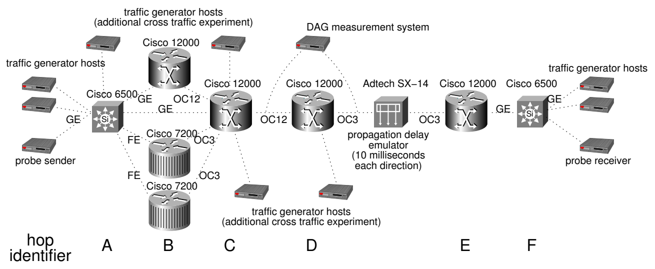


Figure 2. Experimental testbed.

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 to understand where they work well and where they work poorly. 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 Figure 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 [24] (web-like traffic) or Iperf [28] (infinite source and constant bit-rate traffic)[2]. We used an Adtech SX-14 hardware propagation delay emulator configured to add a delay of 10 milliseconds in each direction for all experiments.

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

Topology 1 (narrow and tight link are the same, homogeneous RTT): Probe traffic was configured to cross the GE 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.

Topology 2 (narrow and tight link are not the same, homogeneous RTT): 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[3].

Topology 3 (narrow and tight link are not the same, heterogeneous RTT): Using again web-like cross traffic, we configured our Linux traffic generation hosts with NetPath [5] to emulate round-trip times of 20, 50, 80 and 110 milliseconds. We also attached 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 Endace DAG 3.5 packet capture cards (affording timestamping accuracy on the order of single microseconds [9][4]) 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 § 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 [13, 14, 21]. 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 microseconds.

The phase plots for these experiments shown in Figure 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 microseconds for PATHLOAD packets, 80 microseconds for SPRUCE packets) or with one cross traffic packet interposed (150 microseconds for PATHLOAD, 160 microseconds 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 Figure 3(a) by the perceptible shift to the right in the upper cluster of points. A similar shift exists in Figure 3(b). Finally, we note that points below the diagonal line in Figure 3(a) represent evidence for compression in PATHLOAD streams[5].

To further explore the problem of bias imposed by probe senders, we collected several thousand packet spacing mea-

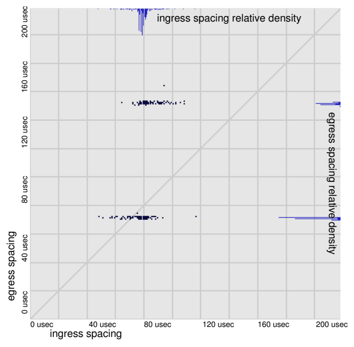
[2]Our traffic generator hosts were identically configured workstations running either Linux 2.4 and FreeBSD 5.3. The workstations had 2 GHz Intel Pentium 4 processors, 2 GB of RAM and Intel Pro/1000 cards (with interrupt coalescence disabled). Each system was dual-homed, so that all management traffic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

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

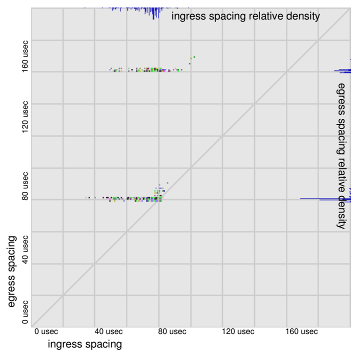
[4]As a consistency and calibration check, we also captured traces using Endace DAG 3.8 cards, which employ a higher frequency clock, and have somewhat different architectural features than the DAG 3.5. The resulting phase plots were consistent with those produced using the DAG 3.5. Experiments described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

[5]We quantify the prevalence of compression in [25]. In general, about 20% of PATHLOAD streams experience compression.

surements from SPRUCE and PATHLOAD and compared each spacing with the spacing measured at the DAG monitor between hops C and D. Figure 4(a) 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.



(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).

Figure 3. Phase plots of PATHLOAD and SPRUCE streams. Grid lines are separated by 20 microseconds for each plot. CBR cross traffic of 50 Mb/s, with uniform UDP packets of 1500 bytes (not shown in plots) causes bimodal output spacing distribution of probe traffic. Target input spacing for each tool is 80 microseconds. Note the slightly different scale for each plot.

Next, we examined the measurements at the receiving application. PATHLOAD timestamps outgoing/incoming packets using the `gettimeofday()` operating system 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. Figure 4(b) 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.

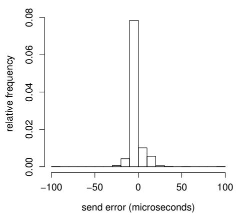
As a final calibration check, and to test whether these results were unique to the hardware and OS configuration used, we attached a DAG 4 (Gigabit Ethernet) monitor directly to the Intel Pro/1000 on a Linux 2.4 workstation and collected additional measurements using SPRUCE. A histogram of differences between spacings measured at SPRUCE and spacings measured at the DAG 4 is shown in Figure 4(c). 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.

4.3. ABET Calibration: Algorithmic Adjustment

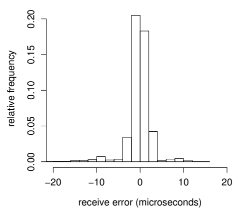
We have found the use of phase plots to be beneficial in our ABET calibration study. They not only helped us to expose end-host limitations of generating precise streams and to identify the resulting bias, but by studying the egress spacings, we were also able to gain an expectation of what the receiving host *should have* measured and realized that considering compression events is important. In summary, phase plot analysis resulted in the following observations: (*i*) the error introduced by end hosts has approximately zero mean when multiple measurements are taken; (*ii*) the relationship between input and output probe rates and AB described in Eq. (2) invites refinements; and (*iii*) 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 at four target spacings of 60, 80, 100, and 120 microseconds, in separate experiments. We ran the tool under topology 1 with no cross traffic to collect approximately 1000 packet stream

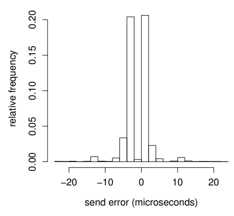
[6]We modified SPRUCE and PATHLOAD to log these timestamps.



(a) Distribution of error at FreeBSD 5.3 sender with Intel Pro/1000 GE adapter and DAG 3.5 between hops C and D.



(b) Distribution of error at FreeBSD 5.3 receiver with Intel Pro/1000 GE adapter and DAG 3.5 between hops D and E.



(c) Distribution of error at Linux 2.4 sender with Intel Pro/1000 GE adapter and DAG 4 attached directly to network interface.

Figure 4. Relative frequencies of error between send or receive packet spacings and spacings measured at DAG monitor.

measurements (*i.e.*, about 100,000 packets per experiment). 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 microsecond. The distributions show that the mean error converges to zero quite quickly and that packet streams of at least length 20 appear to be sufficient. However, 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.

At the base of our proposed algorithm is a version of Equation (2), 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 average difference in input and output spacings as an indication of whether the input rate was above the available bandwidth along the path (analogous to Eq. (2)):

$$|\overline{g_{in}} - \overline{g_{out}}| = \begin{cases} \leq \zeta * \overline{g_{in}} & r_{in} \leq A \\ > \zeta * \overline{g_{in}} & r_{in} > A. \end{cases} \qquad (3)$$

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 to be some minimum value (effectively setting the maximum AB measurable), then proceed as follows.

1. Send probe stream, measuring $\overline{g_{in}}$ and $\overline{g_{out}}$ 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. 3), increase g_{target} by $\frac{|\overline{g_{in}} - \overline{g_{out}}|}{2}$, 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.

We consider the above algorithm to be a "calibrated Pathload" and have implemented it in a tool called YAZ. The source code for YAZ will be available to the research community for evaluation.

4.4. Experimental Evaluation

We compared the accuracy of PATHLOAD, SPRUCE, and YAZ using the different scenarios described in § 4.1. For the CBR and long-lived TCP source experiments, we continuously collected AB estimates from each tool for 10 minutes, discarding the first 30 seconds and last 30 seconds. For the web traffic setups, we continuously collected AB estimates from each tool for 30 minutes, also discarding the first 30 seconds and last 30 seconds. 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. 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 $\approx$ 15 Mb/s.

For all experiments, YAZ was configured with $\alpha = 0.3$ in its exponentially-weighted moving average (this setting produced minimum mean squared error over all experiments) 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[7]. We set YAZ's stream length to 50 packets. For SPRUCE, we use 149.76 Mb/s as the tight link capacity in Equation (1) 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[8]. Additional details of our results are in [25].

Results for all the experiments are shown in Figure 5. The results for constant bitrate traffic in topology 1 (Figure 5(a)) 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. Figure 5(b) 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 (Figure 5(c)), 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). Figure 5(e) shows the result of this experiment, comparing the YAZ and SPRUCE results from Figure 5(d). 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 Figure 5(f). 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, Figure 5(g) 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.

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 2 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.

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.

Table 1. Summary of errors between packet spacings measured at application send and receive, and DAG monitors. All values are in microseconds. Negative values indicate that a larger spacing was measured at DAG monitor than in the application.

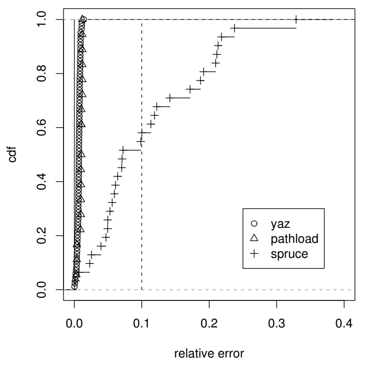
	DAG 3.5/3.8 (OC-3/12) FreeBSD 5.3		DAG 4 (GE) Linux 2.4
	Send Error	Receive Error	Send Error
Min	-93.00	-20.00	-24.00
Median	-2.00	0.00	-2.00
Mean	-1.54	0.15	-0.61
Max	100.00	18.00	23.00

Table 2. 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.

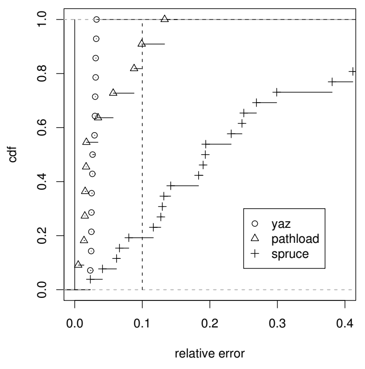
	Estimates Produced	Latency μ (σ) (seconds)	Iterations per Estimate μ (σ)	Mean Pkts per Estimate
PATHLOAD ($K = 100$)	96	17.7 (3.8)	8.4 (4.8)	10080
PATHLOAD ($K = 50$)	97	17.6 (3.8)	8.8 (4.2)	5280
SPRUCE	156	10.9 (0.9)	NA	200
YAZ	446	3.8 (1.5)	6.1 (8.8)	366

[7]Although it is an important calibration task, we omit a detailed analysis of the sensitivity of YAZ to its parameters due to space limitations.

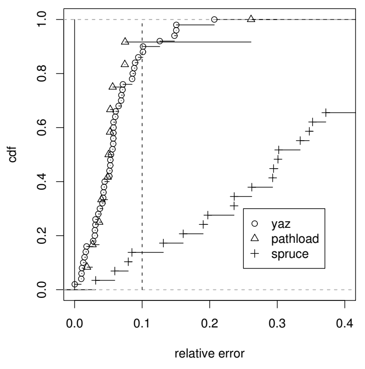
[8]Using the midpoint of its estimation range is, in general, favorable to PATHLOAD [25].



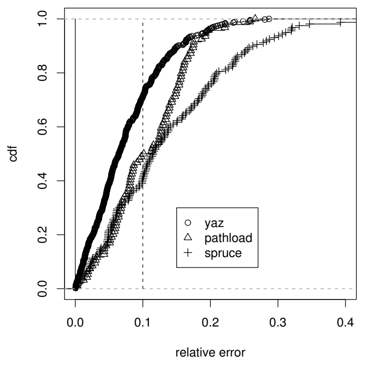
(a) Constant bit rate cross traffic of 50 Mb/s (topology 1).



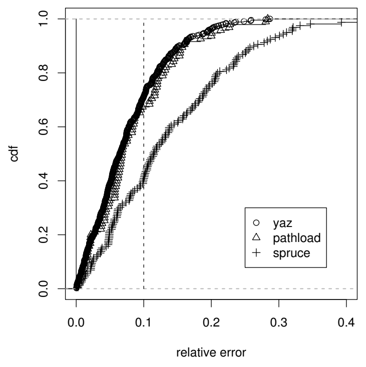
(b) Long-lived TCP sources in one direction (left to right in Figure 2) (Topology 1).



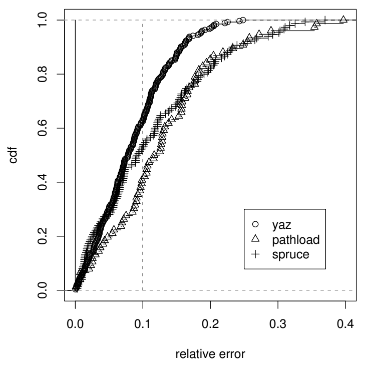
(c) Long-lived TCP sources in two directions (Topology 1).



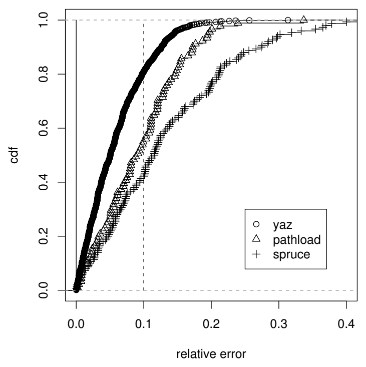
(d) Web-like cross traffic produced by Harpoon with average rate of 50 Mb/s (Topology 1).



(e) Comparison of YAZ, PATHLOAD, and SPRUCE, for web-like traffic when PATHLOAD is configured for streams of length 100 (Topology 1). (YAZ and SPRUCE curves are same as in Figure 5(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).

Figure 5. Comparison of available bandwidth estimation accuracy between YAZ, PATHLOAD, and SPRUCE for six cross 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.

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 are available for use by other researchers [4].

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 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. 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.

References

[1] DETER: A laboratory for security research. http://www.isi.edu/deter/, 2006.

[2] Emulab — network emulation testbed. http://www.emulab.net, 2006.

[3] PLANETLAB — an open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org, 2006.

[4] The Wisconsin Advanced Internet Laboratory. http://wail.cs.wisc.edu, 2006.

[5] S. Agarwal, J. Sommers, and P. Barford. Scalable network path emulation. In *Proceedings of IEEE MASCOTS '05*, September 2005.

[6] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *Proceedings of ACM IMC '03*, October 2003.

[7] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Proceedings of Performance '96*, Lausanne, Switzerland, October 1996.

[8] Y.-C. Cheng, U. Hölzle, N. Cardwell, S. Savage, and G. Voelker. Monkey see, monkey do: A tool for TCP tracing and replaying. In *Proceedings of the USENIX 2004 Conference*, June 2004.

[9] S. Donnelly. *High Precision Timing in Passive Measurements of Data Networks*. PhD thesis, University of Waikato, 2002.

[10] S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4), August 2001.

[11] F. Hernandez-Campos, F. Smith, and K. Jeffay. How real can synthetic network traffic be? In *Proceedings of ACM SIGCOMM '04 (Poster Session)*, 2004.

[12] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 21(6), August 2003.

[13] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, Pensylvania, August 2002.

[14] G. Jin and B. Tierney. System capability effects on algorithms for network bandwidth measurement. In *Proceedings of ACM SIGCOMM Internet Measurement Conference '03*, October 2003.

[15] K. Lakshiminarayanan, V. Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *Proceedings of ACM IMC '04*, Taormina, Sicily, Italy, 2004.

[16] L. Le, J. Aikat, K. Jeffay, and F. Smith. The effects of active queue management on web performance. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, 2003.

[17] S. McCanne and S. Floyd. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[18] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proceedings of Global Internet Symposium*, 2000.

[19] V. Paxson. End-to-end Internet packet dynamics. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[20] V. Paxson. Strategies for sound Internet measurement. In *Proceedings of ACM IMC '04*, October 2004.

[21] R. Prasad, M. Jain, and C. Dovrolis. Effects of interrupt coalescence on network measurements. In *Proceedings of PAM '04*, April 2004.

[22] V. Riberio, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient available bandwidth estimation for network paths. In *Proceedings of Passive and Active Measurement Workshop*, 2003.

[23] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and kc claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proceedings of PAM '05*, 2005.

[24] J. Sommers and P. Barford. Self-configuring network traffic generation. In *Proceedings of ACM SIGCOMM Internet Measurement Conference '04*, 2004.

[25] J. Sommers, P. Barford, and W. Willinger. A proposed framework for calibration of available bandwidth estimation tools (extended version). Technical Report 1546, Computer Sciences Department, University of Wisconsin-Madison, 2005.

[26] J. Sommers, P. Barford, and W. Willinger. SPLAT: A visualization too for mining Internet measurements. In *Proceedings of Passive and Active Measurement Conference*, 2006.

[27] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of ACM IMC '03*, Miami, Florida, October 2003.

[28] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf 1.7.0 – the TCP/UDP bandwidth measurement tool. http://dast.nlanr.net/Projects/Iperf. 2006.