Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links

Alok Shriram[1], Margaret Murray[2], Young Hyun[1], Nevil Brownlee[1], Andre Broido[1], Marina Fomenkov[1], and kc claffy[1]

[1] CAIDA, San Diego Supercomputer Center, University of California, San Diego
{alok,youngh,nevil,broido,marina,kc}@caida.org
[2] Texas Advanced Computing Center
marg@tacc.utexas.edu

Abstract. In this paper we present results of a series of bandwidth estimation experiments conducted on a high-speed testbed at the San Diego Supercomputer Center and on OC-48 and GigE paths in real world networks. We test and compare publicly available bandwidth estimation tools: *abing*, *pathchirp*, *pathload*, and *Spruce*. We also tested *Iperf* which measures achievable TCP throughput. In the lab we used two different sources of known and reproducible cross-traffic in a fully controlled environment. In real world networks we had a complete knowledge of link capacities and had access to SNMP counters for independent cross-traffic verification. We compare the accuracy and other operational characteristics of the tools and analyze factors impacting their performance.

1 Introduction

Application users on high-speed networks perceive the network as an end-to-end connection between resources of interest to them. Discovering the least congested end-to-end path to distributed resources is important for optimizing network utilization. Therefore, users and application designers need tools and methodologies to monitor network conditions and to rationalize their performance expectations.

Several network characteristics related to performance are measured in bits per second: capacity, available bandwidth, bulk transfer capacity, and achievable TCP throughput. Although these metrics appear similar, they are not, and knowing one of them does not generally imply that one can say anything about others. Prasad *et al.* [1] provide rigorous definitions of terms used in the field, survey underlying techniques and methodologies, and list open source measurement tools for each of the above metrics.

By definition [1], end-to-end capacity of a path is determined by the link with the minimum capacity (narrow link). End-to-end available bandwidth of a path is determined by the link with the minimum unused capacity (tight link). In this study our goal is to test and compare tools that claim to measure the available end-to-end bandwidth (Table 1). We did not test tools that measure end-to-end capacity.

Candidate bandwidth estimation tools face increasingly difficult measurement challenges as link speeds increase and network functionality grows more complex. Consider the issue of time precision: on faster links, intervals between packets decrease, rendering packet probe measurements more sensitive to timing errors. The nominal 1 μs resolution of UNIX timestamps is acceptable when measuring 120 μs gaps between 1500 B packets on 100 Mb/s links but insufficient to quantify packet interarrival time (IAT)

Table 1. Available bandwidth estimation tools.

Tool	Author	Methodology
abing	Navratil [2]	packet pair
cprobe	Carter [3]	packet trains
IGI	Hu [4]	SLoPS
netest	Jin [5]	unpublished
pathchirp	Ribeiro [6]	chirp train
pipechar	Jin [7]	unpublished
pathload	Jain [8]	SLoPS
Spruce	Strauss [9]	SLoPS

variations of 12 μs gaps on GigE links. Available bandwidth measurements on high-speed links stress the limits of clock precision especially since additional timing errors may arise due to the NIC itself, the operating system, or the Network Time Protocol (designed to synchronize clocks of computers over a network) [10].

Several other problems may be introduced by network devices and configurations. Interrupt coalescence improves network packet processing efficiency, but breaks end-to-end tools that assume uniform per packet processing and timing [11]. Hidden Layer 2 store-and-forward devices distort an end-to-end tool's path hop count, resulting in calculation errors [12]. MTU mismatches impede measurements by artificially limiting path throughput. Modern routers that relegate probe traffic to a slower path or implement QoS mechanisms may also cause unanticipated complications for end-to-end probing tools. Concerted cooperative efforts of network operators, researchers and tool developers can resolve those (and many other) network issues and advance the field of bandwidth measurement.

While accurate end-to-end measurement is difficult, it is also important that bandwidth estimation tools be fast and relatively unintrusive. Otherwise, answers are wrong, arrive too late to be useful, or the end-to-end probe may itself interfere with the network resources that the user attempts to measure and exploit.

1.1 Related Work

Coccetti and Percacci [13] tested *Iperf*, *pathrate*, *pipechar* and *pathload* on a low speed ($\leq$ 4 Mb/s) 3 or 4 hop topology, with and without cross-traffic. They found that tool results depend strongly on the configuration of queues in the routers. They concluded that in a real network environment, interpreting tool results requires considerable care, especially if QoS features are present in the network.

Strauss *et al.* [9] introduced *Spruce*, a new tool to measure available bandwidth, and compared it to *IGI* and *pathload*. They used SNMP data to perform an absolute comparison on two end-to-end paths that both had their tight and narrow links at 100 Mb/s. They also compared the relative sensitivity of the tools on 400 paths in the RON and PlanetLab testbeds. The authors found that *IGI* performs poorly at higher loads, and that *Spruce* was more accurate than *pathload*.

Hu and Steenkiste [4] explored the packet pair mechanism for measuring available bandwidth and presented two measurement techniques, *IGI* and *PTR*. They tested their methods on 13 Internet paths with bottleneck link capacities of $\leq$ 100 Mb/s and compared the accuracies to *pathload* using the bulk data transmission rate (*Iperf*) as the

benchmark. On some paths all three tools were in agreement with *Iperf*, while on others the results fluctuated. Since the authors did not have any information about true properties and state of the paths, the validity of their results is rather uncertain.

Finally, Ubik *et al.* [14] ran *ABwE* (a predecessor to *abing*) and *pathload* on two paths on the GEANT network, obtaining one month of hourly measurements. The measured paths consisted of at least 10 routers, all with GigE or OC-48 links. Measurements from both tools did not agree. These preliminary results await further analysis.

Our study takes one step further the testing and comparing publicly available tools for available bandwidth estimation. First, we considered and evaluated a larger number of tools than previous authors. Second, we conducted two series of reproducible laboratory tests in a fully controlled environment using two different sources of cross-traffic. Third, we experimented on high-speed paths in real networks where we had a complete knowledge of link capacities and had access to SNMP counters for independent cross-traffic verification. We compare the accuracy and other operational characteristics of the tools, and analyze factors impacting their performance.

2 Testing Methodology

From Table 1 we selected the following tools for this study: *abing*, *pathchirp*, *pathload*, and *Spruce*. For comparison we also included *Iperf* [15] which measures achievable TCP throughput. *Iperf* is widely used for end-to-end performance measurements and has become an unofficial standard [16] in the research networking community.

We were unable to test *cprobe* [3] because it only runs on an SGI Irix platform and we do not have one in our testbed. We did not include *netest* in this study since in our initial tests this tool inconsistently reported different metrics on different runs and different loads. We excluded *pipechar* [7] after tests on 100 Mb/s paths and *IGI* [4] after tests on 1 Gb/s paths since they were unresponsive to variations in cross-traffic.

2.1 Bandwidth Estimation Testbed

In collaboration with the CalNGI Network Performance Reference Lab [17], CAIDA researchers developed an isolated high-speed testbed that can be used as a reference center for testing bandwidth estimation tools. This resource allows us to test bandwidth estimation tools against known and reproducible cross-traffic scenarios and to look deeply into internal details of tools operation. We also attempt to offer remote access to the lab to tool developers wishing to further refine and enhance their tools.

In our current testbed configuration (Figure 1), all end hosts are connected to switches capable of handling jumbo MTUs (9000 B). Three routers in the testbed end-to-end path are each from a different manufacturer. Routers were configured with two separate network domains (both within private RFC1918 space) that route all packets across a single backbone. An OC48 link connects a Juniper M20 router with a Cisco GSR 12008 router, and a GigE link connects the Cisco with a Foundry BigIron 10 router. We use jumbo MTUs (9000 B) throughout our OC48/GigE configuration in order to support traffic flow at full line speed [18].

Bandwidth estimation tools run on two designated end hosts each equipped with a 1.8 GHz Xeon processor, 512 MB memory, and an Intel PRO/1000 GigE NIC card

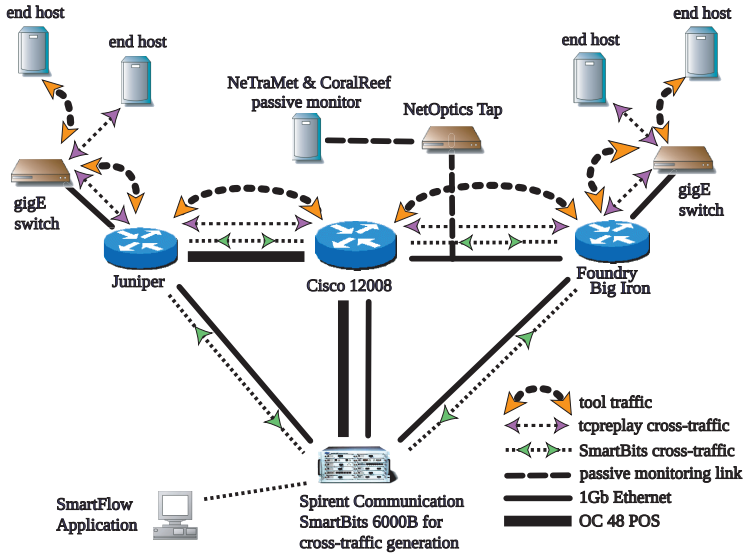


Fig. 1. Bandwidth Estimation Testbed. The end-to-end path being tested traverses three routers and includes OC48 and GigE links. Tool traffic occurs between designated end hosts in the upper part of this figure. Cross-traffic is injected either by additional end hosts behind the jumbo-MTU capable GigE switches or by the Spirent SmartBits 6000 box (lower part of figure). Passive monitors tap the path links as shown to provide independent measurement verification.

installed on a 64b PCI-X 133 MHz bus. The operating system is the CAIDA reference FreeBSD version 4.8.

Our laboratory setup also includes dedicated hosts that run *CoralReef* [19] and *NeTraMet* [20] passive monitor software for independent verification of tool and cross-traffic levels and characteristics. Endace DAG 4.3 network monitoring interface cards on these hosts tap the OC-48 and GigE links under load. *CoralReef* can either analyze flow characteristics and packet IATs in real time or capture header data for subsequent analysis. The *NeTraMet* passive RTFM meter can collect packet size and IAT distributions in real time, separating tool traffic from cross-traffic.

2.2 Methods of Generating Cross-traffic

The algorithms used by bandwidth estimating tools make assumptions about characteristics of the underlying cross-traffic. When these assumptions do not apply, tools cannot perform correctly. Therefore, test traffic must be as realistic as possible with respect to its packet IAT and size distributions.

In our study we conducted two series of laboratory tool tests using two different methods of cross-traffic generation. These methods are described below.

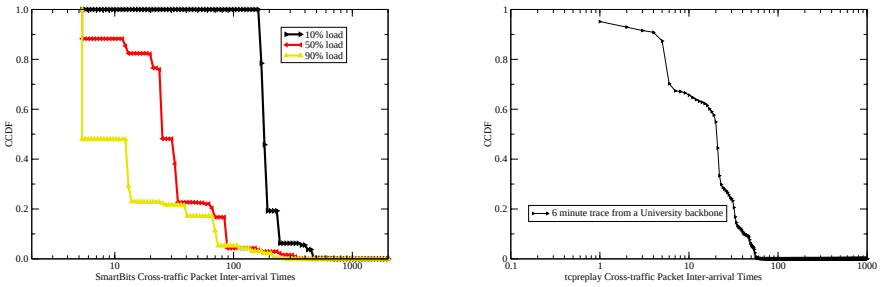
Synthetic Cross-traffic Spirent Communications SmartBits 6000B [21] is a hardware system for testing, simulating and troubleshooting network infrastructure and performance. It uses the Spirent *SmartFlow* [22] application that enables controlled traffic generation for L2/L3 and QoS laboratory testing.

Using SmartBits and *SmartFlow* we can generate pseudo-random, yet reproducible traffic with accurately controlled load levels and packet size distributions. This traffic

generator models pseudo-random traffic flows where the user sets the number of flows in the overall load and the number of bytes to send to a given port/flow before moving on to the next one (burst size). The software also allows the user to define the L2 frame size for each component flow. The resulting synthetic traffic emulates realistic protocol headers. However, it does not imitate TCP congestion control and is not congestion-aware.

In our experiments we varied traffic load level from 100 to 900 Mb/s which corresponds to 10-90% of the narrow GigE link capacity. At each load level, *SmartFlow* generated nineteen different flows. Each flow had a burst size of 1 and consisted of either 64, 576, 1510 or 8192 byte L2 frames. The first three sizes correspond to the most common L2 frame sizes observed in real network traffic [23]. We added the jumbo packet component because high-speed links must employ jumbo MTUs in order to push traffic levels to line saturation. While [23] data suggest a tri-modal distribution of small/medium/large frames in approximately 60/20/20% proportions, we are not aware of equivalent published packet-size data for links where jumbo MTUs are enabled. We mixed the frames of four sizes in equal proportions.

Packet IATs (Figure 2(a)) ranged from 4 to more than 400 μs. We used passive monitors *CoralReef* and *NeTraMet* to verify the actual load level of generated traffic and found that it matched the requirements within 1-2%.



(a) SmartBits (b) tcpreplay

Fig. 2. SmartBits and tcpreplay cross-traffic packet inter-arrival times.

Playing back traces of the real traffic. We replayed previously captured and anonymized traffic traces on our laboratory end-to-end path using a tool *tcpreplay* [24]. This method of cross-traffic generation reproduces actual IAT and packet size distributions but is not congestion-aware. The playback tool operated on two additional end hosts (separate from the end hosts running bandwidth estimation tools) and injected the cross-traffic into the main end-to-end path via GigE switches.

We tested bandwidth estimation tools using two different traces as background cross-traffic:

– a 6-minute trace collected from a 1 Gb/s backbone link of a large university with approximately 300-345 Mb/s of cross-traffic load;

– a 6-minute trace collected from a 2.5 Gb/s backbone link of a major ISP showing approximately 100-200 Mb/s of cross-traffic load.

Neither trace contained any jumbo frames. Packet sizes exhibited a tri-modal distribution as in [23]. Packet IATs (Figure 2(b)) ranged from 1 to 60 μs.

We used CoralReef to continuously measure *tcpreplay* cross-traffic on the laboratory end-to-end path and recorded timestamps of packet arrivals and packet sizes. We converted this information into timestamped bandwidth readings and compared them to concurrent tool estimates. Both traces exhibited burstiness on microsecond time scales, but loads were fairly stable when aggregated over one-second time intervals.

3 Tool Evaluation Results

In this section we present tool measurements in laboratory tests using synthetic, non-congestion-aware cross-traffic with controlled traffic load (*SmartFlow*) and captured traffic traces with realistic workload characteristics (*tcpreplay*). In Section 4 we show results of experiments on real high-speed networks.

3.1 Comparison of Tool Accuracy

Experiments with Synthesized Cross-traffic. We used the SmartBits 6000B device with the *SmartFlow* application to generate bi-directional traffic loads, varying from 10% to 90% of the 1 Gb/s end-to-end path capacity in 10% steps. We tested one tool at a time. In each experiment, the synthetic traffic load ran for six minutes. To avoid any edge effects, we delayed starting the tool for several seconds after initiating cross-traffic and ran the tool continuously for five minutes. Figure 3 shows the average and standard deviation of all available bandwidth values obtained during these 5 minute intervals for each tool at each given load.

Our end-to-end path includes three different routers with different settings. To check whether the sequence of routers in the path affects the tool measurements, we ran tests with synthesized cross-traffic in both directions. We observed only minor differences between directions. The variations are within the accuracy range of the tools and we suspect are due to different router buffer sizes.

We found that *abing* (Figure 3a) reports highly inaccurate results when available bandwidth drops below 600 Mb/s (60% on a GigE link). Note that this tool is currently deployed on the Internet End-to-End Performance Monitoring (IEPM) measurement infrastructure [25] where the MTU size is 1500 B, while our high-speed test lab uses a jumbo 9000 B MTU. We attempted to change *abing* settings to work with its maximum 8160 B probe packet size, but this change did not improve its accuracy.

We looked into further details of *abing* operating on an empty GigE path. The tool continuously sends back-to-back pairs of 1478 byte UDP packets with a 50 ms waiting interval between pairs. *abing* derives estimates of available bandwidth from the amount of delay introduced by the ”network” between the paired packets. *abing* puts a timestamp into each packet, and the returned packet carries a receiver timestamp. Computing the packet IAT does not require clock synchronization since it is calculated as a difference between timestamps on the same host. Since these timestamps have a μs

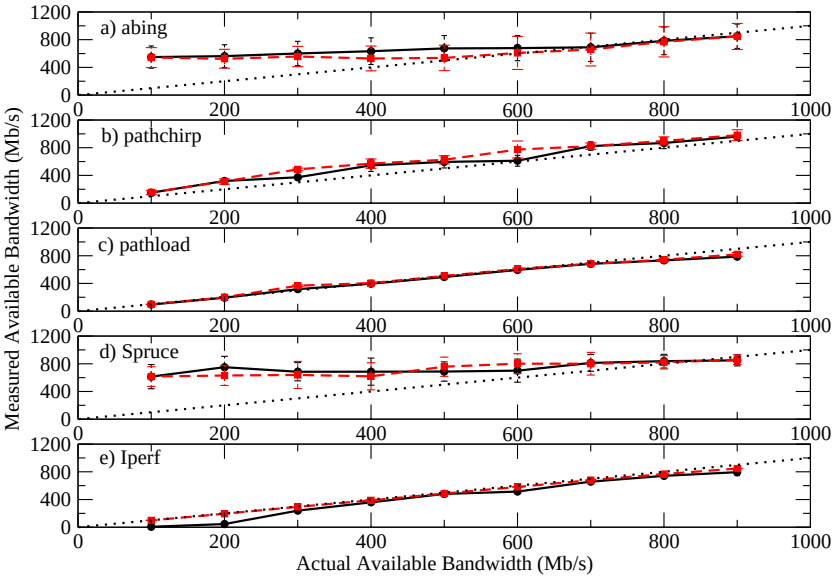


Fig. 3. Comparison of available bandwidth measurements on a 4-hop OC48/GigE path loaded with synthesized cross-traffic. For each experimental point, the x-coordinate is the actual available bandwidth of the path (equal to the GigE link capacity of 1000 Mb/s minus the generated load). The y-coordinate is the tool reading. Measurements of the end-to-end path in both directions are shown. The dash-dotted line shows expected value from SmartBits setting.

granularity, the IAT computed from them is also an integer number of μs. For back-to-back 1500 B packets on an empty 1 Gb/s link (12 Kbits transmitted at 1 ns per bit) the IAT is between 11 and 13 μs, depending on rounding error. However, we observed that every 20-30 packets the IAT becomes 244 μs. This jump may be a consequence of interrupt coalescence or a delay in some intermediate device such as a switch. The average IAT then changes to more than 20 μs yielding a bit rate of less than 600 Mb/s. This observation explains *abing* results: on an empty 1 Gb/s tight link it reports two discrete values of available bandwidth, the more frequent one of 890-960 Mb/s and occasional drops to 490-550 Mb/s. This oscillating behavior is clearly observed in time series of *abing* measurements (Figure 4) described below.

Another tool, *Spruce* (Figure 3d), uses a similar technique and, unsurprisingly, its results are impeded by the same phenomenon. *Spruce* sends 14 back-to-back 1500 B UDP packet pairs with a waiting interval of 160-1400 ms between pair probes (depending on some internal algorithm). In *Spruce* measurements, 244 μs gaps between packet pairs occur randomly between normal 12 μs gaps. Since the waiting time between pairs varies without pattern, the reported available bandwidth also varies without pattern in the 300-990 Mb/s range.

Results of our experiments with *abing* and *Spruce* on high-speed links caution that tools utilizing packet pair techniques must be aware of delay quantization possibly present in the studied network. Also, 1500-byte frames and microsecond timestamp resolution are simply not sensitive enough for probing high-speed paths.

In SmartBits tests, estimates of available bandwidth by *pathchirp* are 10-20% higher than the actual value determined from SmartBits settings (Figure 3b). This consistent overestimation persists even when there is no cross-traffic. On an empty 1 Gb/s path this tool yields values up to 1100 Mb/s. We have as yet no explanation for this behavior.

We found that results of *pathload* were the most accurate (Figure 3c). The discrepancy between its readings and actual available bandwidth was $<$10% in most cases.

The last tested tool, *Iperf*, estimates not the available bandwidth, but the achievable TCP throughput. We ran *Iperf* with the maximum buffer size of 227 KB and found it to be accurate within 15% or better (Figure 4e). Note that a smaller buffer size setting significantly reduces the *Iperf* throughput. This observation appears to contradict the usual rule of thumb that the optimal buffer size is the product of bandwidth and delay, which in our case would be (10^9 b/s) x (10^{-4} s) $\sim$ 12.5 KB. Dovrolis *et al.* discuss this phenomenon in [26].

Experiments with trace playbacks. The second series of laboratory tests used previously recorded traces of real traffic. For these experiments we extracted six-minute samples from longer traces to use as a *tcpreplay* source. As in SmartBits experiments, in order to avoid edge effects we delayed the tool start for a few seconds after starting *tcpreplay* and ran each tool continuously for five minutes.

Figure 4 plots a time series of the actual available bandwidth, obtained by computing the throughput of the trace at a one-second aggregation interval and subtracting that from the link capacity of 1 Gb/s. Time is measured from the start of the trace. We then plot every value obtained by a given tool at the time it was returned.

As described in Section 2.2, we performed *tcpreplay* experiments with two different traces. We present tool measurements of the University backbone trace, which produced a load of about 300 Mb/s leaving about 700 Mb/s of available bandwidth. The tool behavior when using the ISP trace with a load of about 100 Mb/s was similar and is not shown here.

In tests with playback of real traces, *abing* and *Spruce* exhibit the same problems that plagued their performance in experiments with synthetic cross-traffic. Figure 4a shows that *abing* returned one of two values, neither of which was close to the expected available bandwidth. *Spruce* results (Figure 4d) continued to vary without pattern.

pathchirp measurements (Figure 4b) had a startup period of about 70 s when the tool returned only a constant value. The length of this period is related to the tool's measurement algorithm and depends on the number of chirps and chirp packet size selected for the given tool run. After the startup phase, *pathchirp*'s values alternate within 15-20% of the actual available bandwidth.

The range reported by *pathload* (Figure 4c) slightly underestimates the available bandwidth by $<$16%.

Iperf reports surprisingly low results when run against *tcpreplay* traffic (Figure 4e). Two factors are causing this gross underestimation: packet drops requiring retransmission and a too long retransmission timeout of 1.2 s (default value). In the experiment shown, the host running *Iperf* and the host running *tcpreplay* were connected to the main end-to-end path via a switch. We checked the switch's MIB for discarded packets and discovered a packet loss of about 1% when the tool and cross-traffic streams merge. Although the loss appears small, it causes *Iperf* to halve its congestion window and triggers a significant number of retransmissions. The default retransmission timeout is so large it consumes up to 75% of the *Iperf* running time. Decreasing the retransmission timeout to 20 ms and/or connecting the *tcpreplay* host directly to the path bypassing the switch considerably improves *Iperf*'s performance. Note that we were able to reproduce

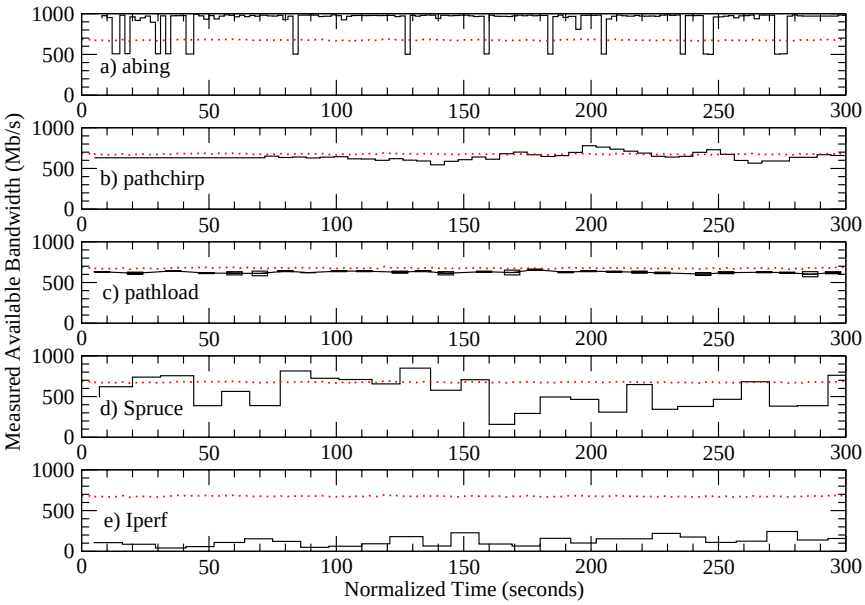


Fig. 4. Comparison of available bandwidth tool measurements on a 4-hop OC48/GigE path loaded with played back real traffic. The X-axis shows time from the beginning of trace playback. The Y-axis is the measured available bandwidth reported by each tool. The dotted line shows the actual available bandwidth that was very stable on a one-second aggregation scale.

the degraded *Iperf* performance in experiments with synthetic SmartBits traffic when we flooded the path with a large number of small (64 B) packets. These experiments confirm that ultimately the TCP performance in the face of packet loss strongly depends on the OS retransmission timer.

3.2 Comparison of Tool Operational Characteristics

We considered several parameters that may potentially affect a user's decision regarding which tool to use: measurement time, intrusiveness, and overhead. We measured all these characteristics in experiments with SmartBits synthetic traffic where we can stabilize and control the load.

We define tool measurement time to be the average measurement time of all runs at a particular load level. On our 4-hop OC-48/GigE topology, the observed measurement durations were: 1.3 s for *abing*, 11 s for *Spruce*, 5.5 s for *pathchirp*, and 10 s for *Iperf* independent of load. The *pathload* measurement time generally increased when the available bandwidth decreased, and ranged between 7 and 22 s.

We define tool intrusiveness as the ratio of the average tool traffic rate to the available bandwidth, and tool overhead as the ratio of tool traffic rate to cross-traffic rate (Figure 3.2). *pathchirp*, *abing*, and *Spruce* have low overhead, each consuming less than 0.2% of the available bandwidth on the GigE link and introducing practically no additional traffic into the network as they measure. *pathload* intrusiveness is between 3 and 7%. Its overhead slightly increases with the available bandwidth (that is, when the cross-traffic actually decreases) and reaches 30% for the 10% load. As expected, *Iperf*

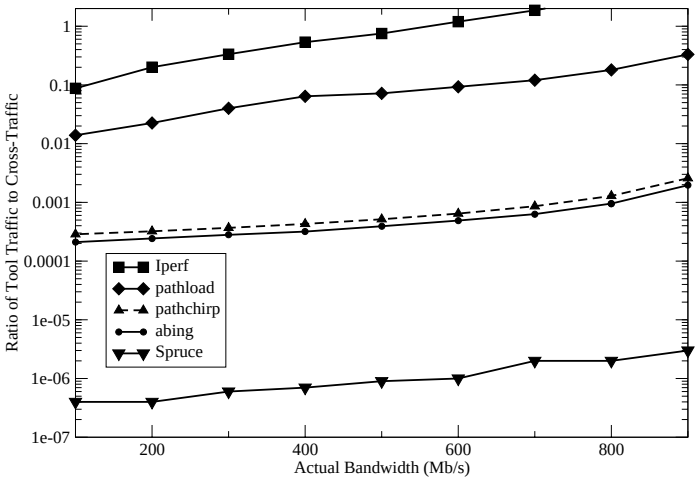


Fig. 5. Tool overhead vs. available bandwidth. Note that *pathchirp*, *abing*, and *Spruce* exhibit essentially zero overhead.

is the most expensive tool both in terms of its intrusiveness (74-79%) and overhead costs. Since it attempts to occupy all available bandwidth, its traffic can easily exceed the existing cross-traffic.

4 Real World Validation

Comparisons of bandwidth estimation tools have been criticized for their lack of validation in the real world. Many factors impede if not prohibit comprehensive testing of tools on production networks. First, network conditions and traffic levels are variable and usually beyond the experimenters' control. This uncertainty prevents unambiguous interpretation of experimental results and renders measurements unreproducible. Second, a danger that tests may perturb or even disrupt the normal course of network operations makes network operators reluctant to participate in any experiments. Only close cooperation between experimenters and operators can overcome both obstacles.

We were able to complement our laboratory tests with two series of experiments in the real world. In both setups, the paths we measured traversed exclusively academic, research and government networks.

Experiments on the Abilene Network. We carried out the available bandwidth measurements on a 6 hop end-to-end path from Sunnyvale to Atlanta on the Abilene Network. Both end machines had a 1 Gb/s connection to the network and sourced no traffic except from running our tools. The rest of links in the path had either 2.5 or 10 Gb/s capacities.

We chose not to test *Spruce* on the Abilene Network since this tool performed poorly in our laboratory experiments[3]. We ran *pathload*, *pathchirp*, *abing*, and *Iperf* for 5 min each, in that order, back-to-back. We concurrently polled the SNMP 64-bit InOctect counters for all routers along the path every 10 s and hence knew the per-link utilization with 10 s resolution. We calculated the per-link available bandwidth as the difference

[3] We tested *Spruce* in the other series of real network experiments, see subsection on SDSC-ORNL paths below

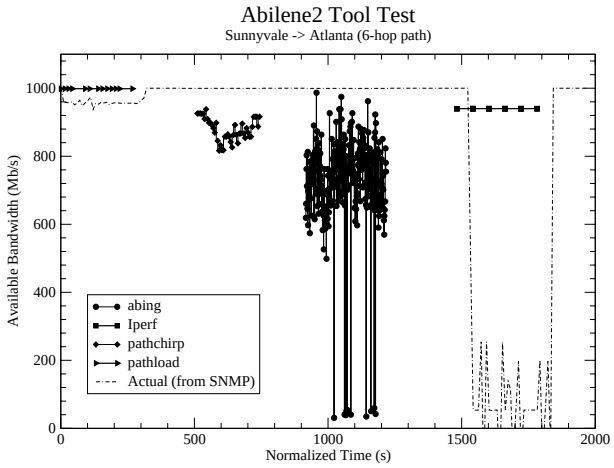


Fig. 6. Real world experiment conducted on the Abilene network. The dashed line shows the available bandwidth derived from SNMP measurements. See explanations in the text.

between link capacity and utilization. The end-to-end available bandwidth is the minimum of per-link available bandwidths. During our experiments, the Abilene network did not have enough traffic on the backbone links to bring their available bandwidth below 1 Gb/s. Therefore, the end machines' 1Gb/s connections were both narrow and tight links in our topology.

Figure 4 shows our tool measurements and SNMP-derived available bandwidth. Measurements with *pathload*, *pathchirp*, and *Iperf* are reasonably accurate, while *abing* readings wildly fluctuate in the whole range between 0 and 1000 Mb/s.

The discrepancy between *Iperf* measurements and SNMP-derived values reflects tool design: *Iperf* generates large overhead ($>$70%) because it intentionally attempts to fill the tight link. Consequent readings of SNMP counters indicate how many bytes traversed an interface of a router during that time interval. They report total number of bytes without distinguishing tool traffic from cross-traffic. If a tool's overhead is high, then available bandwidth derived from SNMP data during this tool run is low. At the same time, since tools attempt to measure available bandwidth ignoring their own traffic, a high-overhead tool will report more available bandwidth than SNMP. Therefore, *Iperf* shows a correct value of achievable TCP throughput of $\sim$950 Mb/s while concurrent SNMP counters account for *Iperf*'s own generated traffic, and thus yield less than 200 Mb/s of available bandwidth. A smaller discrepancy between *pathload* and SNMP results reflects *pathload*'s overhead ($\sim$10% per our lab tests).

Experiments on SDSC-ORNL paths. In the second series of real-world experiments we tested *abing*, *pathchirp*, *pathload*, and *Spruce* between our host at SDSC (running CAIDA reference FreeBSD version 4.8) and a host at Oak Ridge National Lab (running Red Hat Linux release 9 with a 2.4.23 kernel and Web100 patch [27]). These experiments are of limited value since we did not have concurrent SNMP data for comparison with our results. However, we had complete information about link capacities along the paths which at least allows us to distinguish plausible results from impossible ones. We include these experiments since they present first insights into the interplay between the probing packet size and the path MTU.

The two paths we measured are highly asymmetric. The SDSC-ORNL path crosses CENIC and ESNet, has a narrow link capacity of 622 Mb/s (OC12) and MTU of 1500 bytes. The ORNL-SDSC path crosses Abilene and CENIC, has a narrow link capacity of 1 Gb/s and supports 9000-byte packets end-to-end. Both paths remained stable over the course of our experiments and included OC12, GigE, 10 GigE, and OC192 links. Under most traffic scenarios, it seems highly unlikely for the 10 Gb/s links to have less than 1 Gb/s of available bandwidth. Lacking true values of available bandwidth from SNMP counters for absolute calibration of tool results, we assume that the narrow link is also the tight link in both our paths.

Table 2. Summary of wide-area bandwidth measurements ("f"= produced no data).

Direction	Path Capacity, MTU	Probe Packet Size	Tool readings (Mb/s)			
			abing[a]	pathchirp	pathload	Spruce
SDSC to ORNL	622 Mb/s (OC12), 1500	1500	178 / 241	543	>324	296
		9000	f / 664	f	409 – 424	0
ORNL to SDSC	1000 Mb/s (GigE), 9000	1500	727 / 286	807	>600	516
		9000	f / 778	816	846	807

[a] Sender at SDSC for 1st value and at ORNL for 2nd value.

We ran each tool using either 1500 or 9000 byte packets. *abing*, *pathchirp*, and *pathload* support large probe packet size as an option[4]. *Spruce* uses a hardcoded packet size of 1500 bytes; we had to trivially modify the code to increase the packet size to 9000 B. Table 2 summarizes our results while a detailed description is available in [28].

abing has a sender module on one host and a reflector module on the other host and measures available bandwidth in both directions at once. We found that its behavior changed when we switched the locations of sender and reflector. *abing* with 9000 B packets did not return results from SDSC to ORNL ("f" in Table 2). We could see that the ORNL host was receiving fragmented packets, but the *abing* reflector was not echoing packets. In the opposite direction, from ORNL to SDSC, *abing* with 9000 B packets overestimates the available bandwidth for the OC12 path (reports 664 Mb/s on 622 Mb/s capacity). Note that almost the factor of 3 difference in GigE path measurements with 1500 B packets (727 and 286 Mb/s) may be due to different network conditions since these tests occurred on different days.

pathchirp produced results on both paths when run with 1500 B packets and on the GigE path with 9000 B packets, but failed on the OC12 path with large packets. There does not appear to be any significant advantage to using large packets over small ones. Variations between consequent measurements with the same packet size are sometimes greater than the difference between using large and small packets.

In tests with 1500 B packets, on both paths *pathload* reports that results are limited by the maximum host sending rate. With 9000 B packets, this tool yielded available bandwidth estimates for both paths, but issued a warning "actual probing rate [does not equal] desired probing rate" for the OC12 path.

[4] The *abing* reflector has a hardcoded packet size of 1478 bytes.

Spruce performed poorly in experiments with small packets from SDSC to ORNL, reporting wildly fluctuating values of available bandwidth. Tests with 9000 B packets in this direction always produced 0 Mb/s. However, in the ORNL to SDSC direction, its readings were more consistent and on par with other tools.

We suspect that fragmentation is responsible for most of the problems when probing packet size and path MTU mismatch. While using large packets to measure high-speed links is beneficial, more work is necessary to consistently support large packets and to reduce failures and inaccuracies stemming from fragmentation.

5 Conclusions and Future Work

Our study is the first comprehensive evaluation of publicly available tools for available bandwidth estimation on high-speed links. We conducted testing in the lab and over research networks. We found that *pathload* and *pathchirp* are the most accurate tools under conditions of our experiments.

Iperf performs well on high-speed links if run with its maximum buffer window size. Even small ($\sim$1%) but persistent amounts of packet loss seriously degrade its performance. Too conservative settings of the OS retransmission timer further exacerbate this problem.

Results of our experiments with *abing* and *Spruce* caution that tools utilizing packet pair techniques must be aware of delay quantization possibly present in the studied network. Also, 1500 byte frames and microsecond timestamp resolution are not sensitive enough for probing high-speed paths.

Despite the revealed problems, experimenting with available bandwidth estimating tools using large packets is worthwhile, considering the importance of using large packets on high-speed links.

We demonstrated how our testbed can be used to evaluate and compare end-to-end bandwidth estimation tools against reproducible cross-traffic in a fully controlled environment. Several bandwidth estimation tool developers have taken advantage of our offer of remote access to the testbed to conduct their own tests. We plan to use what we have learned from our testing methodology to conduct monitoring efforts on both research and commodity infrastructure.

6 Acknowledgments

We gratefully acknowledge access to the Spirent 6000 network performance tester and Foundry BigIron router in the CalNGI Network Performance Reference Lab created by Kevin Walsh. Many thanks to Cisco Systems for the GSR12008 router, Juniper Networks for the M20 router, and Endace, Ltd. for access to their DAG4.3GE network measurement card. Nathaniel Mendoza, Grant Duvall and Brendan White provided testbed configuration and troubleshooting assistance. Feedback from remote testbed users Jiri Navratil, Ravi Prasad, and Vinay Ribeiro was helpful in refining test procedures. Aaron Turner provided us with a lot of support on installing and running *tcpreplay*. We are grateful to Matthew J Zekauskas for invaluable assistance with running experiments on the Abilene network. This work was supported by DOE grant DE-FC02-01ER25466.

References

1. Prasad, R., Murray, M., claffy, k., Dovrolis, C.: Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In: IEEE Network. (2003)
2. Navratil, J.: ABwE: A Practical Approach to Available Bandwidth. In: PAM. (2003)
3. Carter, R., Crovella, M.: Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report 96-006, Boston University (1996)
4. Hu, N., Steenkiste, P.: Evaluation and Characterization of Available Bandwidth Probing Techniques. IEEE JSAC Internet and WWW Measurement, Mapping, and Modeling (2003)
5. Jin, G.: netest-2 (2004) http://www-didc.lbl.gov/NCS/netest.html.
6. Ribeiro, V.: pathChirp: Efficient Available Bandwidth Estimation for Network Path. In: PAM. (2003)
7. Jin, G., Yang, G., Crowley, B.R., Agarwal, D.A.: Network Characterization Service (NCS). Technical report, LBNL (2001)
8. Jain, M., Dovrolis, C.: Pathload: an available bandwidth estimation tool. In: PAM. (2002)
9. Strauss, J., Katabi, D., Kaashoek, F.: A measurement study of available bandwidth estimation tools. In: IMW. (2003)
10. Pasztor, A., Veitch, D.: On the Scope of End-to-end Probing Methods. Communications Letters, IEEE **6(11)** (2002)
11. Prasad, R., Jain, M., Dovrolis, C.: Effects of Interrupt Coalescence on Network Measurements. In: PAM. (2004)
12. Prasad, R., Dovrolis, C., Mah, B.: The effect of layer-2 store-and-forward devices on per hop capacity estimation. In: IMW. (2002)
13. Coccetti, F., Percacci, R.: Bandwidth measurements and router queues. Technical Report INFN/Code-20 settembre 2002, Instituto Nazionale Di Fisica Nucleare, Trieste, Italy (2002) http://ipm.mib.infn.it/bandwidth-measurements-and-router-queues.pdf.
14. Ubik, S., Smotlacha, V., Simar, N.: Performance monitoring of high-speed networks from the NREN perspective. In: TERENA Networking Conference, Rhodes, Greece. (2004) http://staff.cesnet.cz/ ubik/publications/2004/terena2004.pdf.
15. NLANR: Iperf v1.7.0 (2004) http://dast.nlanr.net/Projects/Iperf.
16. Cottrell, L., Logg, C.: Overview of IEPM-BW Bandwidth Testing of Bulk Data Transfer. In: Sc2002: High Performance Networking and Computing. (2002)
17. San Diego Supercomputer Center : CalNGI Network Performance Reference Lab (NPRL) (2004) http://www.calngi.org/about/index.html.
18. Jorgenson, L.: Size Matters: Network Performance on Jumbo Packets. In: Joint Techs Workshop Columbus, OH. (2004)
19. Keys, K., Moore, D., Koga, R., Lagache, E., Tesch, M., Claffy, k.: The architecture of CoralReef: an Internet traffic monitoring software suite. In: Passive and Active Network Measurement (PAM) Workshop, Amsterdam, Netherlands. (2001)
20. Brownlee, N.: NeTraMet 5.0b3 (2004) http://www.caida.org/tools/measurement/netramet/.
21. Spirent Corp.: Smartbits 6000B (2004) http://spirentcom.com/analysis/view.cfm?P=141.
22. Spirent Corp.: Smartflow (2004) http://spirentcom.com/analysis/view.cfm?P=119.
23. NLANR: Passive Measurement Analysis Datacube (2004) http://pma.nlanr.net/Datacube/.
24. Turner, A.: tcpreplay 2.2.2 - a tool to replay saved tcpdump files at arbitrary speed (2004) http://tcpreplay.sourceforge.net/.
25. SLAC: Internet End-to-end Performance Monitoring - Bandwidth to the World (IEPM-BW) Project (2004) http://www-iepm.slac.stanford.edu/bw/.
26. Dovrolis, C., Prasad, R., Jain, M.: Socket Buffer Auto-Sizing for High-Performance Data Transfers. Journal of Grid Computing **1(4)** (2004)
27. Mathis, M.: The Web100 Project (2003) http://www.web100.org.
28. Hyun, Y.: Running Bandwidth Estimation Tools on Wide-Area Internet Paths (2004) http://www.caida.org/projects/bwest/reports/tool-comparison-supplement.xml.