A Measurement Study of Available Bandwidth Estimation Tools [*]

Jacob Strauss
jastr@mit.edu

Dina Katabi
dk@mit.edu

Frans Kaashoek
kaashoek@mit.edu

MIT Computer Science and Artificial Intelligence Laboratory

ABSTRACT

Available bandwidth estimation is useful for route selection in overlay networks, QoS verification, and traffic engineering. Recent years have seen a surge in interest in available bandwidth estimation. A few tools have been proposed and evaluated in simulation and over a limited number of Internet paths, but there is still great uncertainty in the performance of these tools over the Internet at large.

This paper introduces Spruce, a simple, light-weight tool for measuring available bandwidth, and compares it with two existing tools, IGI and Pathload, over 400 different Internet paths. The comparison focuses on accuracy, failure patterns, probe overhead, and implementation issues. The paper verifies the measured available bandwidth by comparing it to Multi-Router Traffic Grapher (MRTG) data and by measuring how each tool responds to induced changes in available bandwidth.

The measurements show that Spruce is more accurate than Pathload and IGI. Pathload tends to overestimate the available bandwidth whereas IGI becomes insensitive when the bottleneck utilization is large.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Monitoring

General Terms

Measurement, Experimentation, Performance

Keywords

Available bandwidth

[*]This research was conducted as part of the IRIS project (`http://project-iris.net/`), supported by the National Science Foundation under Cooperative Agreement No. ANI-0225660, along with support from the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare Systems Center, San Diego, under contract N66001-00-1-8933.

1. INTRODUCTION

Recent years have seen a strong interest in techniques for estimating available bandwidth along an Internet path. The path diversity in overlay networks creates a need for estimating the available bandwidth over these paths as a method for choosing the best route. Further, in an overlay, one can assume the cooperation of both the sender and the receiver, which is necessary for most probing techniques. Many available bandwidth estimation tools have emerged such as Pathload [12], TOPP [18], PTR/IGI [9], Delphi [21], and Pathchirp [22].

This paper introduces Spruce, a simple tool for estimating available bandwidth, and compares it with two existing tools Pathload, and IGI. In comparison with previous work, this paper provides the first wide-scale Internet experiments to evaluate current tools for measuring available bandwidth. Reported experiments with IGI, Pathload, and TOPP have been limited to a few Internet-wide paths [9, 12, 18]. In contrast, our measurements involve 400 different Internet-wide paths. We have collected our measurements using PlanetLab nodes [2] and the RON testbed [4], targeting paths with a variety of capacity, hop count, latency, load, and link technology.

The *available bandwidth (ABW)* at a link is its unused capacity. (See Figure 1 for the definitions used in this paper.) Since, at any time, a link is either idle or transmitting packets at the maximum speed, the definition of the available bandwidth ought to look at the average unused bandwidth over some time interval T. Thus,

$$A_i(t,T) = \frac{1}{T}\int_t^{T+t} (C_i - \lambda_i(t))\,dt, \tag{1}$$

where $A_i(t,T)$ is the available bandwidth at link i at time t, C_i is the link's capacity, and λ_i is its traffic. The available bandwidth along a path is the minimum available bandwidth of all traversed links.

Spruce (Spread PaiR Unused Capacity Estimate) is a tool for end hosts to measure available bandwidth. It samples the arrival rate at the bottleneck by sending pairs of packets spaced so that the second probe packet arrives at a bottleneck queue before the first packet departs the queue. Spruce then calculates the number of bytes that arrived at the queue between the two probes from the inter-probe spacing at the receiver. Spruce computes the available bandwidth as the difference between the path capacity and the arrival rate at the bottleneck.

Similarly to IGI [9] and Delphi [21], Spruce is designed around the probe gap model (see §2), which assumes a sin-

Term	Definition
Capacity	The maximum rate at which packets can be transmitted by a link
Narrow link	The link with the smallest capacity along a path
Available bandwidth	A link's unused capacity
Tight link	The link with minimum available bandwidth along a path

Figure 1: The definitions for the terms: capacity, available bandwidth, narrow link and tight link. The tight link may be different from the narrow link along the path. For example, consider a two-link path that traverses a T1 link with 1.5 Mb/s and a 10 Mb/s Ethernet. The narrow link along this path is the T1. But, it is quite possible that the Ethernet is more congested and has less unused bandwidth than the T1, in which case, the tight link is the 10 Mb/s Ethernet.

gle bottleneck. However, our experiments show that Spruce works well in realistic environments and is robust against deviations from this assumption.

The results of the measurements on the PlanetLab and RON testsbeds can be summarized as follows:

- Almost 70% of Spruce's measurements had a relative error smaller than 30%. Pathload and IGI experienced larger errors.
- Pathload consistently over- or under-estimated the available bandwidth, whereas IGI did not respond properly to injected cross traffic and overestimated available bandwidth on some paths.
- Pathload generated between 2.5 and 10 MB of probe traffic per measurement. In contrast, the average per-measurement probe traffic generated by IGI is 130 KB and that generated by Spruce is 300 KB.

2. DESIGN SPACE & RELATED WORK

Keshav's work on packet pair [14] is the earliest attempt to estimate the available bandwidth using measurements conducted at the end hosts. Packet pair assumes Fair Queuing in the routers and as a result cannot estimate the available bandwidth in the current Internet.

Cprobe [6] is a pioneering tool for estimating the available bandwidth using end-to-end measurements. Cprobe doesn't assume fair queueing. Instead of using a pair of packets, cprobe sends a short train of ICMP packets and computes the available bandwidth as the probe traffic divided by the interval between the arrival of the last ICMP ECHO and the first ICMP ECHO in the train. A similar approach is used by pipechar [13]. Dovrolis et al [7] show that these techniques measure a metric called the Asymptotic Dispersion Rate (ADR), which is related to the available bandwidth but not the same.

The recent set of tools can be distinguished according to the two main approaches underlying the estimation techniques.

- **The probe gap model (PGM)** exploits the information in the time gap between the arrivals of two successive probes at the receiver. A probe pair is sent with a time gap Δ_{in}, and reaches the receiver with a time gap Δ_{out}. Assuming a single bottleneck and that the queue does not become empty between the departure of the first probe in the pair and the arrival of the second probe, then Δ_{out} is the time taken by the bottleneck to transmit the second probe in the pair and the cross traffic that arrived during Δ_{in}, as shown in Figure 2. Thus, the time to transmit the cross traffic is $\Delta_{out} - \Delta_{in}$, and the rate of the cross-traffic is $\frac{\Delta_{out}-\Delta_{in}}{\Delta_{in}} \times C$, where C is the capacity of the bottleneck. The available bandwidth is:

$$A = C \times \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}\right). \qquad (2)$$

  Spruce, IGI [9], and Delphi [21] are example tools that use the gap model.

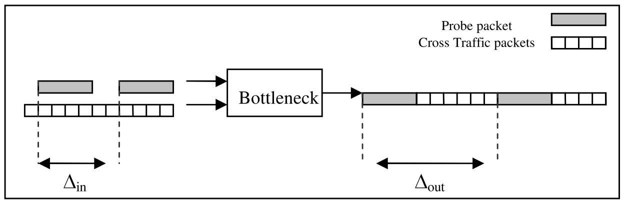


Figure 2: The Probe Gap Model (PGM) for estimating available bandwidth.

- **The probe rate model (PRM)** is based on the concept of self-induced congestion; informally, if one sends probe traffic at a rate lower than the available bandwidth along the path, then the arrival rate of probe traffic at the receiver will match their rate at the sender. In contrast, if the probe traffic is sent at a rate higher than the available bandwidth, then queues will build up inside the network and the probe traffic will be delayed. As a result, the probes' rate at the receiver will be less than their sending rate. Thus, one can measure the available bandwidth by searching for the turning point at which the probe sending and receiving rates start matching. Tools such as Pathload [12], Pathchirp [22], PTR [9], and TOPP [18] use the probe rate model.

To cope with the burstiness of cross traffic, both the PGM and PRM tools use a train of probe packets to generate a single measurement.

Both the PGM and PRM approaches assume: 1) FIFO queuing at all routers along the path; 2) cross traffic follows a fluid model (i.e., non-probe packets have an infinitely small packet size); 3) average rates of cross traffic change slowly and is constant for the duration of a single measurement. Further, the probe gap model assumes a single bottleneck which is both the narrow and tight link for that path. These assumptions are necessary for the model analysis but the tools might still work even when some of the assumptions do not hold [9].

The literature is rich in related work that does not directly estimate the available bandwidth. Paxson defines the relative available bandwidth metric β, which indicates the degree of congestion along the path but does not directly provide an estimate of the available bandwidth. Pathchar [10], bprobe [6], pchar [5], tailgating [15], nettimer [16], clink [8], pathrate [7] are tools for estimating capacity. Treno [17], and cap [3] estimate the TCP fair rate along a path.

Finally, Zhang et al. [23] examined stationarity of TCP throughput measurements over many Internet paths. They found that in many cases, TCP rates varied by less than a factor of three over the course of an hour or more. Paxson [20] found that routes between Internet hosts are often stable on scales ranging from hours to days. These two results are important because they indicate that we can repeat experiments back to back and expect similar results.

3. SPRUCE

Spruce is based on the probe gap model (PGM) described in §2. Like other PGM tools [21, 9], Spruce assumes a single bottleneck that is both the narrow and tight link along the path (see definitions in Figure 1). The results from Internet measurements reported in §4 and §5 show that Spruce is fairly accurate in realistic Internet settings, where this assumption might not hold.

3.1 Design

Spruce computes the available bandwidth according to Equation 2 (see §2), which requires 3 parameters: C, Δ_{in}, and Δ_{out}. Spruce assumes C is known, sets Δ_{in} at the sender, and measures Δ_{out} at the receiver.

At the sender, Spruce sets the intra-pair time gap, Δ_{in}, to the transmission time of a 1500B data packet on the bottleneck link. This choice ensures that the queue does not empty between the departures of the two probe packets in a pair, which is a requirement for Equation 2.

At the receiver, Spruce measures Δ_{out}, the transmission time of both cross traffic and a 1500B probe. With this information and a known capacity for the bottleneck link, Spruce then calculates the number of bytes that arrived at the queue between the two probes in a pair from the inter-probe spacing as $\frac{\Delta_{out}-\Delta_{in}}{\Delta_{in}} \times C$, where C is the capacity of the bottleneck. Plugging these numbers into Eq. 2, gives Spruce one sample measurement of the available bandwidth.

To improve accuracy of the estimate, Spruce performs a sequence of probe-pair measurements and reports the average. Spruce sets the inter-gap time between two probe pairs to the output of an exponentially distributed function, whose average τ is much larger than Δ_{in}, resulting in a Poisson sampling process. This decision is appealing for two reasons. First, for a simple model which assumes a single bottleneck and non-fluid cross-traffic (i.e., no cross traffic or close to capacity cross traffic), a sequence of measurements according to a Poisson sampling process sees the average cross traffic rate.

Second, Poisson sampling ensures that Spruce is non-intrusive. In particular, sending a sequence of packet pairs instead of a packet train allows us to control the inter-pair gap independently from the intra-pair gap. We use a large inter-pair gap τ to make Spruce non-intrusive. Other tools which send packet trains at high peak rates may disturb concurrent TCP flows even though each train is of short duration. Spruce computes the available bandwidth at time t as the average of the last K sample measurements. The default value for K is 100.

3.2 Implementation

Spruce consists of separate user-level sender and receiver programs. The sender takes as arguments the DNS name of the receiver, and the known capacity of the path. We have tested Spruce on Linux 2.4.19 and FreeBSD 4.7 systems.

The Spruce sender sends a series of pairs of 1500B UDP packets. Spruce sets the intra-pair gap to the transmission time of a 1500B packet on the path's narrow link. The sender adjusts the average inter-pair gap to ensure that the probe rate is the minimum of 240Kb/s and 5% of the path capacity. For example, on a 1.5 Mb/s path, the average inter-pair gap is set to 320 ms, resulting in a probe rate of 75 Kb/s.

Since the gaps between two packets in a pair can be small, the sender program reads the system clock in a polling loop, not releasing the processor voluntarily until the pair has been transmitted. If the operating system reschedules the sender program between two packets of a pair, the program, when it receives the processor again, gives up sending the second packet, and restarts.

The receiving kernel timestamps each received packet using the SO_TIMESTAMP socket option. The spruce receiver calculates Δ_{out} using the timestamps, and computes an estimate of the available bandwidth on that path using Eq. 2. The receiver averages individual samples using a sliding window over 100 packets.

3.3 Spruce Characteristics

The following properties distinguish Spruce from other available bandwidth tools.

1. Spruce uses a Poisson process of packet pairs rather than packet trains (or chirps). This form of sampling allows Spruce to be both non-intrusive and robust, as explained in §3.1.

2. By carefully choosing the value of Δ_{in}, Spruce ensures that the bottleneck queue does not empty between the two probes in a pair, which is a requirement for the correctness of the gap model.

3. Spruce separates capacity measurement from available-bandwidth measurement. It assumes that capacity can be measured easily with one of the capacity measurement tools and that capacity stays stable when measuring available bandwidth. For the environments for which Spruce is designed, selecting paths in overlay networks, this assumption holds.

4. Spruce doesn't overwhelm the narrow link on a path, because its probe rate is no more traffic than the minimum of 240 Kb/s and 5% of the capacity of the narrow link.

5. Apart from the number of pairs K over which to average the measurements, Spruce does not have any tunable parameters.

4. ABSOLUTE ACCURACY

We evaluate the ability of Pathload, IGI, and Spruce to compute the available bandwidth in real network settings.

We focus on these three tools because they cover the spectrum of underlying models in this area; Pathload is a pure PRM tool; Spruce is a pure PGM tool; whereas IGI borrows from both models. IGI first finds the turning point at which the probes' sending rate starts matching their receiving rate. Then it sends a train of packets at that rate and computes the available bandwidth using the probe gap information.

4.1 Methodology: The MRTG Test

The Multi-Router Traffic Grapher (MRTG) [19] reports the amount of traffic forwarded by a router interface. It collects its measurements from the router's Management Information Base (MIB) using SNMP, generating a reading every 5 minutes. Given the capacity of the link, the MRTG data allows us to compute the average available bandwidth every 5 minutes. Despite its low resolution, the MRTG data is the most accurate way to verify the output of available bandwidth estimation tools.

This method requires access to MRTG logs from all links along the path and the knowledge of the capacity of all traversed links. Because of these difficulties, we apply the MRTG test only to a pair of paths for which we have such data. One path, which traverses MIT's campus network, consists of 5 hops, has an RTT of 4 ms, and its tight and narrow link is a 100 Mb/s Ethernet connecting the Lab of Computer Science (LCS) to the rest of the MIT network. The other path is from UC Berkeley to MIT over the Abilene network, with 17 hops and an RTT of 76 ms. This path also has a tight and narrow link of 100 Mb/s, though the remainder of the path is 1 Gb/s or faster. While we do not claim that these paths are representative of most Internet paths, they may be typical of many university networks.

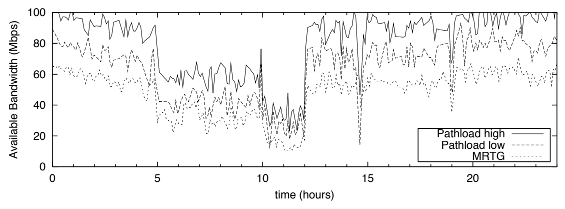
We monitor these paths over a period of one week and for a total monitoring time of 130 hours. We repeatedly run Pathload followed by IGI then Spruce. All three tools use the same sender and receiver machines. Since MRTG data provides an average over a period of 5 minutes, we smooth the measurements by taking the average output of each tool over similar 5 minute periods.

Occasionally, we actively increase the cross traffic traversing the monitored path. The objective of this induced load is to discover the responsiveness of the measurement tools to changing network conditions. The sender of the cross traffic is different from the machine running the tools. The cross traffic uses UDP (though similar results were obtained with TCP cross traffic). We generate cross traffic by taking a few NLANR [1] traces and playing them at an adjustable rate, while maintaining the same packet size. This choice ensures that the packet size distribution of our cross traffic follows the size distribution in the Internet.

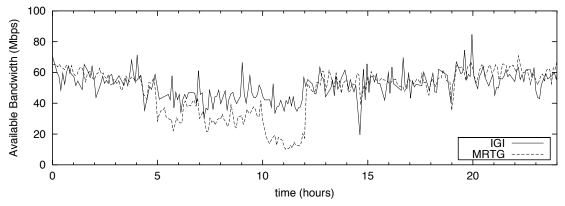
4.2 MRTG Test Results

Figures 3 and 4 illustrate typical segments of our results. They plot the available bandwidth over a period of a day as measured by MRTG, Pathload, IGI, and Spruce. In figure 3, during the interval from hour 5 to 10, we inject cross traffic at a rate of 20 Mb/s. From hour 10 to 12, we increase the generated cross traffic rate to 40 Mb/s. The rest of the time, we only monitored the path using the various tools. In figure 4 we inject cross traffic at 20 Mb/s from hour 10 to 14.

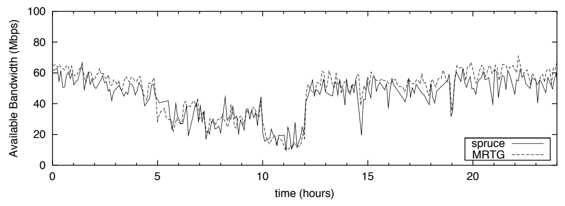
The main observation from Figures 3 and 4 is that *Pathload was consistently inaccurate, over- or under-estimating the available bandwidth, whereas IGI did not respond prop-*



(a)Pathload

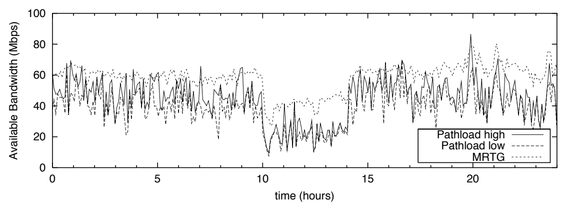


(b)IGI

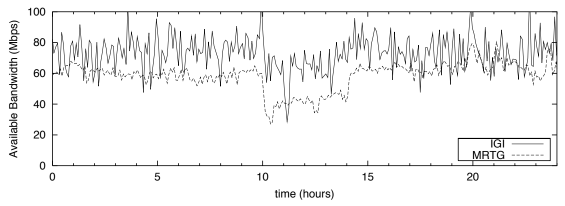


(c)Spruce

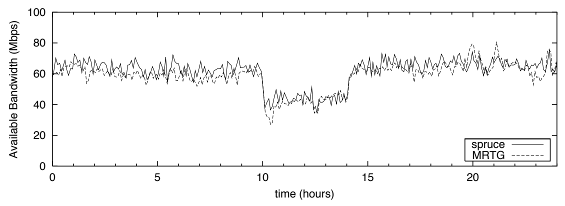
Figure 3: MRTG available bandwidth estimate vs. Pathload, IGI, and Spruce. Data is for a 100 Mb/s path connecting LCS to the MIT network.



(a)Pathload



(b)IGI



(c)Spruce

Figure 4: MRTG available bandwidth estimate vs. Pathload, IGI, and Spruce. Data is for the 100 Mb/s path connecting UC Berkeley to MIT LCS.

erly to injected cross traffic and overestimated available bandwidth on some paths. In contrast, Spruce tracked the available bandwidth reasonably well in all cases. This behavior was repeatedly observed in the MRTG tests.

Figure 3(a) shows that Pathload's upper and lower bounds on the available bandwidth are both too high. Indeed, sometimes the lower bound is 20 Mb/s higher than the actual available bandwidth as measured by MRTG. Despite its overestimation of the available bandwidth, Pathload reacts properly to the cross traffic injected in the interval $[5, 12]$. To ensure that these inaccurate bounds on available bandwidth were not caused by a bug in the new Pathload release (version 1.1.0), we ran the same experiment with the older version of Pathload which was used in [11], but the behavior persisted. A close examination of the logs show that, on the shorter path, Pathload repeatedly overestimates the turning point at which the probe train/stream starts showing an increasing delay trend, indicating that the probe rate has exceeded the available bandwidth. A preliminary investigation shows that the default values for the Pathload parameters S_{PCT} and S_{PDT} are too high for this path. This reason could be why our results are different from those reported in the Pathload paper [12]. In Figure 4(a) we observed the opposite error, with Pathload detecting a turning point below the true available bandwidth.

Figure 3(b) shows that IGI successfully estimates the available bandwidth when the link utilization is low. However, IGI reacts little to the cross traffic injected during the interval $[5, 12]$. In Figure 4(b), IGI consistently overestimates available bandwidth. We hypothesize that IGI performs poorly when utilization is high, which is consistent with the data reported in Figure 12 of the IGI paper [9]. To some extent, this inaccuracy can be explained based on the IGI algorithm. IGI starts with an initial phase to determine the turning point at which $\Delta_{in} = \Delta_{out}$. Unfortunately, when the utilization is high, this turning point becomes unpronounced, immersed in measurement noise [18].

Finally, Figures 3(c) and 4(c) show that Spruce closely tracks the average available bandwidth and correctly responds to the injected cross traffic. We believe that Spruce's good performance is due to its simplicity and the lack of tunable parameters. Neither of these two paths contain distinct tight and narrow links, nor does either consist of multiple bottlenecks of equal capacity. We expect that Spruce would not perform as well under such conditions.

The explanations of the observed behavior of Pathload and IGI are preliminary. More experiments are needed to better understand the behavior of these tools.

5. RELATIVE ACCURACY

We evaluate Pathload, IGI, and Spruce over a variety of Internet paths using PlanetLab nodes [2] and the RON testbed [4]. We explore over 400 different paths with a variety of capacity, hop count, latency, load, and link technology. In the absence of MRTG data, there is no way to discover the true available bandwidth along a path during the experiment. To overcome this limitation, we use a differential test (D-test) that measures changes in the available bandwidth rather than absolute values.

5.1 Methodology: The D-Test

The D-test has two phases. First, we run the tool and read its estimate of the available bandwidth, M_1. Second, we inject a stream of cross traffic whose rate is $\Gamma = 0.5 \times M_1$, and run the tool again. The induced cross traffic is generated as described in §4.1. Assuming the traffic of other users does not change much between the two phases, the correct change in the available bandwidth is Γ, which is what we expect the tool to estimate in the second phase. Admittedly, the test relies on the assumption that apart from our generated cross traffic, the network conditions do not change between the two phases. We believe this limitation is intrinsic to any evaluation technique that does not have direct access to the routers. To mitigate the impact of changing network conditions on our results, we conduct the two phases very close to each other (both finish within a minute). Further, we repeat the same experiment multiple times over the same path, check for consistency, and ignore outliers.

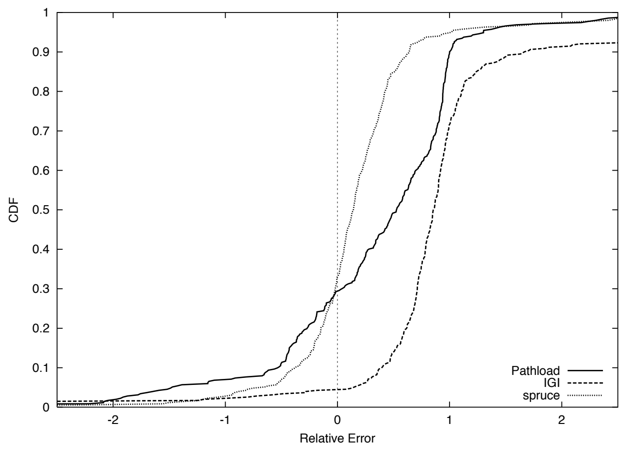


Figure 5: Cumulative distribution of relative error measured by Pathload, IGI, and Spruce across all tested paths in the RON and PlanetLab testbeds.

Each single experiment consists of 3 D-tests over the same path, one with Pathload, second with IGI, and third with Spruce. The tests run one after the other and complete within 2 to 3 minutes.

To measure accuracy we use the *relative error* defined as:

$$Relative\ Error = \frac{\Gamma - (M_1 - M_2)}{\Gamma}, \tag{3}$$

where Γ is the induced change in cross traffic, and M_1 and M_2 are the available bandwidth measured by the same tool in the first and second phases of the same D-test.

5.2 D-Test Results

Figure 5 shows the cumulative distribution function (CDF) of the relative error of Pathload, IGI, and Spruce. The Pathload available bandwidth estimate is computed as the average of the high and low estimates (i.e., $\frac{R_{max}+R_{min}}{2}$).

Ideally, the CDF should be a step function at "0", which means that all experiments resulted in zero errors. In practice, all three tools show some errors and their CDFs are far from ideal. A negative relative error means that the tool has underestimated the available bandwidth whereas a positive relative error means that the tool has overestimated the available bandwidth. The region between "-1" and "1" refers to experiments in which the tool has correctly detected a decrease in available bandwidth between phase 1 and phase 2 of the D-test, but has potentially underestimated or overestimated the change in ABW.

The results in Figure 5 agree with the results of the MRTG test illustrated in Figures 3 and 4. First, the Spruce CDF has a closer shape to a step at "0" than the CDFs of the other tools, indicating that Spruce is more accurate than IGI and Pathload. In particular, almost 70% of Spruce's measurements have a relative error smaller than 30% (the region in which $-0.3 <$ relative error < 0.3). Second, Pathload is less accurate when compared to Spruce because its CDF is further away from a step function at "0". In comparison with IGI, Pathload is more responsive because its CDF ramps up at lower relative error rates. Third, the IGI CDF is almost a step function at "1", which means that IGI reacted very little to charges in the available bandwidth between phase 1 and phase 2 of the D-test. This behavior agrees with the results of the MRTG test which show that IGI is insensitive to induced cross traffic.

Many of the paths in this study have multiple narrow links with equal capacity. Although Spruce is based on the PGM approach which assumes a single bottleneck, it does perform reasonably well in this environment.

Finally, we have computed the average per-measurement probe traffic generated by the various tools. We found that Pathload generates between 2.5 and 10 MB of probe traffic per measurement. In contrast, the average per-measurement probe traffic generated by IGI is 130 KB and that generated by Spruce is 300 KB.

6. IMPLEMENTATION ISSUES

All of the studied tools (i.e., IGI, Pathload, and Spruce) require careful scheduling of probe traffic. More precisely, the input gap between a pair of probes must be accurate and, sometimes, as small as a few hundred microseconds. Because processes cannot sleep for intervals shorter than one kernel tick (10ms or 1ms are common values), each tool uses a delay loop that holds the processor until either preempted or done sending a train of packets. Because this delay loop effectively blocks all other programs from sending traffic for the duration of an entire train, the tools cannot properly measure cross traffic sent from the machine on which it runs. Low bandwidth paths (T1, DSL) are an exception to this rule since the input gap, in this case, is large enough for the tool to release the CPU between packets. Any application that uses one of these measurement tools must therefore use some other mechanism to account for the effect of traffic sent from the same machine on available bandwidth estimates. Applications could either account for their own traffic explicitly, or operating systems could provide scheduling methods to send packets at precise intervals without holding the processor for the duration of a packet train.

7. CONCLUSION

This paper introduced Spruce, a tool for estimating available bandwidth and compared its performance with two existing tools, IGI and Pathload. Spruce is simple, and generates a relatively low amount of probe traffic. Experiments over a large number of Internet paths indicate that Spruce estimates available bandwidth more accurately than Pathload and IGI. Our future work will investigate the sources of observed errors in order to improve the accuracy of available bandwidth estimation. Source code for Spruce will be available at `http://project-iris.net/`.

8. REFERENCES

[1] National Laboratory for Applied Network Research (NLANR). `http://www.nlanr.net/`.

[2] Planetlab. `http://www.planet-lab.org/`.

[3] M. Allman. Measuring End-to-End Bulk Transfer Capacity. In *ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco, CA, Nov. 2001.

[4] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[5] B.A.Mah. pchar: a Tool for Measuring Internet Path Characteristics. Feb. 1999.

[6] R. L. Carter and M. E. Crovella. Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks. Technical Report TR-96-007, Boston University Computer Science Department, 1996.

[7] C. Dovrolis, P. Ramanathanm, and D. Moore. What Do Packet Dispersion Techniques Measure? In *IEEE INFOCOM'01*, 2001.

[8] A. B. Downey. Using Pathchar to Estimate Internet Link Characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.

[9] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 2003.

[10] V. Jacobson. Pathchar. `ftp://ftp.ee.lbl.gov/pathchar/`.

[11] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In *ACM SIGCOMM*, Pittsburg, PA, 2002.

[12] M. Jain and C. Dovrolis. Pathload: A Measurement Tool for End-to-End Available Bandwidth. In *Passive and Active Measurements*, Fort Collins, CO, March 2002.

[13] G. Jin, G. Yang, B. Crowley, and D. Agarwal. Network Characterization Service (NCS). In *the 10th IEEE Symposium on High Performance Distributed Computing*, Aug 2001.

[14] S. Keshav. A Control-Theoretic Approach to Flow Control. In *ACM SIGCOMM '91*, pages 3–15, September 1991.

[15] K. Lai and M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. In *ACM SIGCOMM*, pages 283–294, 2000.

[16] K. Lai and M. Baker. Nettimer: A tool for Measuring Bottleneck Link Bandwidth. In *USENIX Symposium on Internet Technologies and Systems*, March 2001.

[17] M. Mathis. TReno Bulk Transfer Capacity. draft-ietf-ippm-treno-btc-03.txt (Internet-Draft Work in progress).

[18] B. Melander, M. Bjorkman, and P. Gunningberg. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In *Global Internet Symposium*, 2000.

[19] T. Oetiker and D. Rand. Multi Router Traffic Grapher. `http://people.ee.ethz.ch/~oetiker/webtools/mrtg/`.

[20] V. Paxson. End-to-end Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[21] V. J. Ribeiro, M. Coates, R. H. Riedi, S. Sarvotham, and R. G. Baraniuk. Multifractal cross traffic estimation. In *Proc. of ITC specialist seminar on IP traffic Measurement*, September 2000.

[22] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Passive and Active Measurement Workshop*, 2003.

[23] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.