IMR-Pathload: Robust Available Bandwidth Estimation under End-Host Interrupt Delay

Seong-Ryong Kang and Dmitri Loguinov

Texas A&M University, College Station, TX 77843, USA
{skang,dmitri}@cs.tamu.edu

Abstract. Many paths in PlanetLab cannot be measured by Pathload. One of the main reasons for this is timing irregularities caused by interrupt moderation of network hardware, which delays generation of interrupts for a certain period of time to reduce per-packet CPU overhead. Motivated by this problem, we study Pathload in detail under various end-host interrupt delays and find that its trend detection mechanism becomes susceptible to non-negligible interrupt delays, making it unable to measure network paths under such conditions. To overcome this, we propose a new method called IMR-Pathload (*Interrupt Moderation Resilient Pathload*), which incorporates robust trend detection algorithms based on signal de-noising techniques and reliably estimates available bandwidth of network paths under a wide range of interrupt delays. Through experiments in Emulab and Internet, we find that IMR-Pathload substantially improves Pathload's measurement reliability and produces accurate bandwidth estimates under a variety of real-life conditions.

Key words: Bandwidth estimation, network measurement, interrupt moderation, and interrupt delays

1 Introduction

Bandwidth of Internet paths is an important metric for applications. Extensive research has been conducted over the years and the vast majority of work in this area focuses on end-to-end measurement. Although several techniques [4], [13], [11], [12], [14] attempt to measure capacity of the narrow link (i.e., the slowest link in a path) or both capacity and available bandwidth of the tight link (i.e., link with the smallest available bandwidth over a path), many measurement techniques and public tools (such as [6], [9], [16]) have been developed to estimate available bandwidth of the tight link. These methods mainly focus on fast estimation with high accuracy under a various traffic conditions. However, since the ultimate goal of bandwidth estimators is to measure diverse Internet paths, before being a full-blown measurement tool, it is highly desirable that tools are resilient to timing irregularities caused by various OS scheduling delay jitter or hardware interrupt moderation in real networks.

Note that to accurately measure bandwidth, all existing methods heavily rely on high-precision delay measurement of probe packets at end-hosts. However, irregular timing due to interrupt moderation at network interface cards (NICs)

has been identified as the major problem of existing bandwidth estimation tools in practice [15]. To reduce the effect of interrupt moderation, recent tools such as Pathchirp [16] and Pathload described in [15] incorporate mechanisms that aim to "weed out" packets affected by interrupt delays. However, Pathchirp requires manual modification to force it to send (often substantially) more probing packets to obtain an accurate estimate, prolonging measurement undesirably. On the other hand, Pathload attempts to filter out affected packets without increasing the number of probing packets, which unfortunately has a limited effect when interrupt delays become non-trivial. This makes Pathload's estimation much more susceptible to error, which happens fairly often in practice.

To address the above filtering problem without increasing measurement duration, we investigate Pathload's internal algorithm and find that its estimation instability with non-negligible interrupt delays stems from its delay-trend detection mechanism that is not robust under bursty packet arrival introduced by network hardware. To overcome this, we introduce two trend-detection algorithms based on signal de-noising techniques such as wavelet decomposition and window-based averaging and call the new method IMR-Pathload (*Interrupt Moderation Resilient Pathload*). Through experiments in Emulab [5] under various network settings, we find that IMR-Pathload significantly improves Pathload's performance in a wide range ($0 - 500$ μs) of interrupt delays δ. Especially, under non-trivial interrupt delays (e.g., $\delta > 125$ μs), while Pathload fails to produce estimates for any of the paths studied in this paper, IMR-Pathload measures their available bandwidth with over 88% accuracy. Internet experiments also confirm that IMR-Pathload reliably produces bandwidth estimates even for the paths that are not measurable by Pathload.

2 Related Work

A number of techniques have been proposed to measure available bandwidth of network paths [6], [9], [16], which sends N back-to-back packets and discover a relationship between sending rates at the sender and the corresponding receiving rates at the receiver to produce bandwidth estimates of the paths. Among them, we discuss two promising tools that use mechanisms to mitigate the effect of interrupt moderation.

Pathchirp [16] uses packet-trains (called chirps) with exponentially decreasing inter-packet spacings in each chirp and infers available bandwidth using the queuing delay signature of arriving chirps. The basic idea behind this method is that when a transmission rate r_k of a packet k in a chirp reaches available bandwidth of a path under consideration, then subsequent packets $j > k$ in the chirp will exhibit increasing queueing delay. Hence, available bandwidth of the path is the rate r_k of the packet k whose queueing delay starts increasing. To overcome the packet-timing problem introduced by end-host interrupt moderation, Pathchirp increases the number of probing packets in each chirp by a manually selected amount and uses only those packets that (ideally) have not been affected by interrupt delays.

Different from Pathchirp, Pathload [9] sends a fleet of packet-trains with a fixed rate and adjusts the sending rate for the next fleet based on delay-trend information provided by the receiver. Pathload searches for an available bandwidth range by increasing or decreasing the sending rate of probe-trains in a binary search fashion according to trend information. Although Pathload can reduce the effect of interrupt delays without increasing the number of packets in each probe-train, its algorithm is effective only under small interrupt delays.

3 Issues of Interrupt Delay in Bandwidth Measurement

As use of interrupt moderation has become a common practice in modern network settings, host machines in real networks may employ interrupt delays that vary widely in order to reduce CPU utilization and to increase network throughput. It is reported in [7] that the range of interrupt delays recommended for Intel Gigabit NIC (GbE) is $83 - 250$ μs for Microsoft Windows-based systems and $125 - 1000$ μs for Linux-based systems. Jin *et al.* [10] also report that a variety of systems equipped with Gigabit NICs require to delay generation of interrupts over 470 μs to achieve good throughput in receiving high-speed TCP streams and to substantially reduce CPU utilization. The question we have now is how this wide range of interrupt delays affects Pathload's bandwidth estimation. We discuss this issue next.

3.1 Impact of Interrupt Delay

To investigate the potential impact of interrupt moderation on Pathload, we conduct experiments in Emulab [5] for different interrupt delays at the receiver[1]. We start by describing the experimental setup.

Experimental Setup For this investigation, we use a topology shown in Fig. 1, in which source PS sends probe data to the destination PR through five routers $R_1 - R_5$. Nodes S_i $(i = 1, 2, 3, 4)$ send cross-traffic packets to destination nodes D_i at an average rate λ_i. The speed of all access links is 100 Mb/s (delay 10 ms) and the remaining links L_i $(i = 1, 2, 3, 4)$ between routers R_i and R_{i+1} have capacities C_i and propagation delay 40 ms.

To examine Pathload's estimation reliability, we use six different network settings shown in Table 1, which lists the capacity C_i and available bandwidth A_i of each link L_i for different experimental scenarios. The shaded values in each row represent the tight-link capacity and available bandwidth of the path for each case. The values in square brackets represent the capacity of the narrow link (i.e., bottleneck bandwidth) for each case.

In all experiments, we use TCP cross-traffic generated by Iperf traffic generators [8] to load network paths. For this purpose, we run 100 threads in each cross-traffic source S_i to generate TCP flows that are injected into routers R_1,

[1] In Emulab, users can change configuration of network cards.

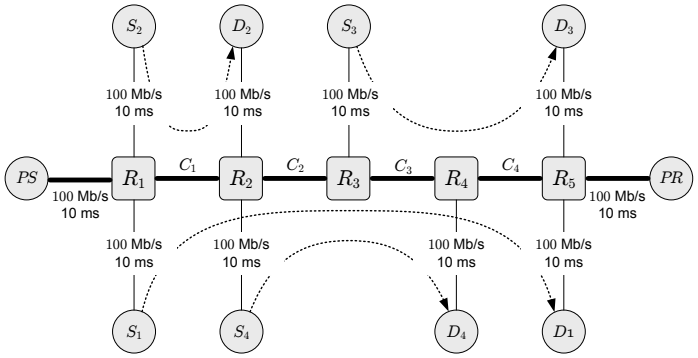


Fig. 1. Evaluation topology in Emulab.

Table 1. Evaluation Setup

Experimentation	Different link bandwidths (Mb/s)							
scenarios	C_1	A_1	C_2	A_2	C_3	A_3	C_4	A_4
Case-I	75	31.84	90	51.69	90	42.05	[60]	40.77
Case-II	75	41.32	90	70.76	90	46.77	[60]	26.39
Case-III	[60]	35.88	90	70.76	[90]	23.39	75	18.10
Case-IV	[60]	21.60	90	65.99	90	42.07	75	36.72
Case-V	[60]	50.25	90	61.17	90	41.99	75	50.86
Case-VI	75	28.97	90	37.8	90	13.86	[60]	31.22

R_2, and R_3 and keep the utilization of each router R_i according to the values shown in Table 1. To maintain a fixed average utilization at each link in experiments, we place an additional (auxiliary) router (not shown in the figure) between node S_1 and router R_1, S_2 and R_1, S_3 and R_3, and S_4 and R_2 to limit the aggregate sending rate of the TCP flows to the capacity of the additional router. The utilization of R_i is controlled by properly setting the capacity of the auxiliary router.

Estimation Reliability Using the above setup, we run Pathload with 4 different values of interrupt delays δ. To demonstrate estimation accuracy, we define the following relative error metric: $e_A = |A - \tilde{A}|/A$, where A is the true available bandwidth of a path and $\tilde{A}$ is its estimate. We report estimation results for each case in Table 2, which show relative estimation errors e_A of available bandwidth. As the table shows, with relatively small interrupt delays (e.g., $\delta \leq 100$ μs), Pathload estimates available bandwidth of the tight link with over 80% accuracy for all cases studied in this paper. Note, however, from the table that when δ becomes larger than 125 μs, it is unable to produce estimates for any of the cases as shown in the table as empty cells, which suggests that its algorithm is susceptible to non-trivial interrupt delays. We also conduct experiments with $\delta = 250$ and 500 μs and confirm its inability, but omit these results for brevity.

Table 2. Pathload's Measurement in Emulab

Interrupt delay δ	Evaluation scenario					
	Case-I	Case-II	Case-III	Case-IV	Case-V	Case-VI
0 μs	9.45%	8.00%	7.57%	6.48%	16.58%	15.01%
100 μs	1.44%	8.52%	14.9%	5.74%	3.6%	20.74%
125 μs	$--$	$--$	15.01%	$--$	$--$	34.65%
$>$ 125 μs	$--$	$--$	$--$	$--$	$--$	$--$

Next, we investigate Pathload's internal algorithm in detail and identify what causes its measurement to be unstable under non-negligible values of δ.

3.2 Analysis

Recall that Pathload [9] sends back-to-back packets in a train of size $N = 100$ with a fixed rate R and examines one-way delay[2] (OWD) of each packet in the probe-train in order to identify a trend exists in the time-series delay data. Based on OWD delay trend, Pathload determines whether the current rate R is faster than the available bandwidth of the path under investigation. Hence, proper detection of OWD trend in a probe-train is crucial for it to produce an accurate and reliable bandwidth estimate of the path.

Note that Pathload first perform ADR (Asymptotic Dispersion Rate) probing by sending a single packet-train and checks interrupt moderation, which it detects when more than 60% of packets in a probe-train have been received back-to-back (with zero or negligible inter-packet delay). If interrupt moderation is detected, Pathload first eliminates such coalesced packets from the received train. Then, it directly performs PCT (Pairwise Comparison Test) and PDT (Pairwise Difference Test) on the remaining data if the number of remaining packets is no less than 5. Recall that the PCT metric represents the fraction of consecutive OWD pairs that are increasing, while the PDT metric quantifies how strong the difference between the first and last OWDs in the data set is. Define X_j to be the one-way delay of a packet j in a set of size n. Then, the PCT and PDT metrics[3] are given by [9]:

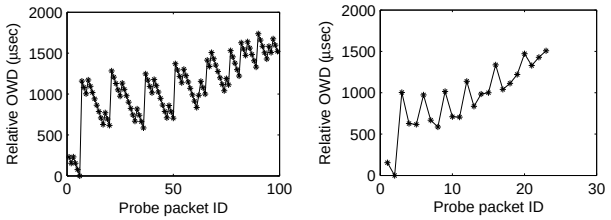
$$PCT = \frac{1}{n-1}\sum_{j=2}^{n} I(X_j > X_{j-1}), \quad PDT = (X_n - X_1)/\sum_{j=2}^{n} |X_j - X_{j-1}|, \quad (1)$$

where $I(Y)$ is one if Y holds, zero otherwise.

On the other hand, when Pathload does not detect interrupt moderation from the initial check, it first eliminates back-to-back packets from the probe-train just like the previous case. If the number of remaining packets is no smaller

[2] One-way delay of a packet is defined as the difference between its arrival time at the receiver and the corresponding sending time at the sender.

[3] Pathload [9] determines OWDs as "increasing" if $PCT > 0.66$, "non-increasing" if $PCT < 0.54$, or "ambiguous" otherwise. Similarly, it identifies OWDs as "increasing" if $PDT > 0.55$, "non-increasing" if $PDT < 0.45$, or "ambiguous" otherwise.



(a) OWD (original) (b) Sampled OWD

Fig. 2. Relative OWDs obtained using the path in case I ($A = 31$ Mb/s).

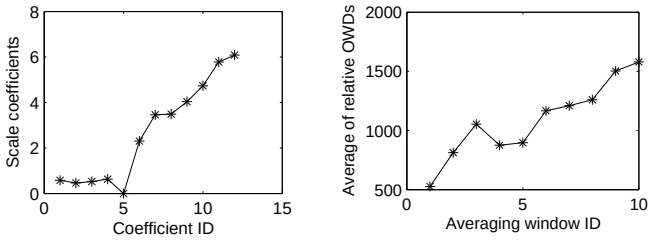
than 36, then Pathload selects OWDs from the remaining packets using median-based sampling (see [9] for details) and applies the PCT and PDT tests to the sampled OWDs.

To assess Pathload's trend detection mechanism, we conduct experiments for Case I with interrupt delay $\delta = 250$ μs. In this example, we collect one-way delay data by running Pathload with a fixed rate $R = 38$ Mb/s and examine how its internal algorithm specifies a delay-trend existing in OWDs. Fig. 2(a) illustrates relative OWDs (one-way delays subtracted by their minimum value) obtained by sending packet trains at 38 Mb/s over the path in case I (available bandwidth $A = 31$ Mb/s). Note in the figure that OWDs exhibit an increasing trend over all even though they decrease in a small-scale burst (successive OWDs in the same burst decrease if the latency for transferring a packet from NIC to the user space at the receiver is smaller than the inter-packet dispersions exiting NIC at the sender [15]). Since the PCT and PDT tests cannot accurately detect a trend present in this kind of coalesced data, Pathload first removes coalesced packets before applying the PCT and PDT tests. Fig. 2(b) shows remaining OWDs after eliminating the coalesced packets. However, even with the data shown in Fig. 2(b), Pathload is unable to detect the increasing trend present in the data since its trend-test produces $PCT = 0.5$ and $PDT = 0.11$. This indicates that Pathload's trend-detection mechanism is not robust under the presence of coalesced packets due to interrupt delays.

Note that Pathload often discards entire packet-trains even with strong presence of a trend in the data due to its inability to detect the trend accurately. Although more extensive evaluations are required to confirm our findings, we believe that Pathload's inaccuracy in trend detection is the major problem that makes it unlikely to be successful in real networks.

4 IMR-Pathload

Motivated by the difficulty of characterizing delay variations in measured noisy OWD data, we study noise-filtering techniques such as wavelet-based signal processing and window-based averaging and explore their applicability in reliably identifying a trend from the data. In what follows below, we first investigate



(a) Wavelet (b) Average

Fig. 3. Wavelet coefficients and 10-packet window averages of relative OWDs shown in Fig. 2(a).

wavelet-based signal processing techniques that are widely used in removing noise from various data sets obtained empirically [2]. To overcome the effect of interrupt delays on trend detection, we apply a simple multi-level discrete wavelet transform [1] to OWDs before performing PCT- and PDT-based trend-test.

Note that in the multi-level wavelet decomposition, each stage consists of scale and wavelet filters followed by down-sampling by a factor of 2 and separates an input signal into two sets of coefficients: scale and wavelet coefficients. The wavelet coefficients represent a noise component in the input signal and thus are not processed further. On the other hand, the scale coefficients are applied to the two filters in the next level as an input to further reduce noise that might still exist in the scale coefficients from the previous stage. As a decomposition level increases, the frequency of wavelets used in filters decreases, capturing lower frequency components present in the original signal.

For experiments in this section, we use the family of Daubechies wavelets [3], which are well known standard wavelets (other wavelets can be used, but performance comparison among different wavelets is beyond the scope of this paper). Specifically, we use Daubechies' length-4 wavelets, whose scale filter coefficients are given by $h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, and $h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$, while its wavelet filter coefficients are $g_0 = h_3$, $g_1 = -h_2$, $g_2 = h_1$, and $g_3 = -h_0$.

Assume that a sequence $s_0, s_1, \ldots, s_{n-1}$ is an input to the j-th stage filters. Define $cA_{j,k}$ and $cD_{j,k}$ (where $k = 0, 1, \ldots, n/2$) to be the scale and wavelet coefficients produced at level j, respectively. Then, $cA_{j,k}$ and $cD_{j,k}$ are given by:

$$cA_{j,k} = h_0 s_{2k} + h_1 s_{2k+1} + h_2 s_{2k+2} + h_3 s_{2k+3} \tag{2}$$

$$cD_{j,k} = g_0 s_{2k} + g_1 s_{2k+1} + g_2 s_{2k+2} + g_3 s_{2k+3}. \tag{3}$$

Note that when $k \geq n/2 - 1$, there are not enough data in the input sequence to compute the coefficients using (2) and (3). This is known as a boundary condition [17], which requires a special treatment that adds more data points to the input sequence (in this paper, we add the last value if necessary).

To demonstrate the effect of wavelet decomposition on trend detection, we decompose OWDs shown in Fig. 2(a) up to level 3 and plot in Fig. 3(a) the scale coefficients that represent the trend component of OWD data. Applying the

Table 3. Emulab Experiment

Estimation method	Interrupt delay δ	Evaluation scenario					
		Case-I	Case-II	Case-III	Case-IV	Case-V	Case-VI
IMR-Pathload (wavelet)	0 μs	2.46%	1.23%	3.47%	2.69%	3.71%	6.52%
	100 μs	6.47%	4.5%	3.02%	4.42%	5.98%	12.17%
	125 μs	7.21%	2.64%	3.88%	1.32%	6.1%	10.77%
	500 μs	5.12%	2.17%	6.78%	3.24%	7.23%	5.56%
IMR-Pathload (average)	0 μs	2.07%	2.24%	2.1%	2.18%	9.67%	5.05%
	100 μs	0.19%	0.71%	11.69%	1.32%	4.19%	6.82%
	125 μs	1.44%	1.82%	12.58%	1.59%	2.64%	7.89%
	500 μs	4.43%	4.59%	9.27%	2.55%	8.95%	6.48%

same PCT and PDT tests to the scale coefficient data, we get $PCT = 0.75$ and $PDT = 0.78$, which means that OWDs exhibit an increasing trend according to the criteria used in Pathload (recall that Pathload fails to detect this increasing trend as discussed in §3.2).

Next, we explore how window-based averaging improves trend detection in noisy data. In this approach, we take the average of OWDs in a window of size k (k-packet sliding window). Using a smaller window makes trend-detection susceptible to a larger interrupt delay since it may not sufficiently remove noise from OWDs (we leave optimal selection of window size as future work). For this example, we employ $k = 10$ and plot in Fig. 3(b) 10-packet window averages of relative OWDs shown in Fig. 2(a), which clearly shows an increasing trend. With these averaged OWDs, we get $PCT = 0.8$ and $PDT = 0.74$, which leads us to conclude that an increasing trend exists in the measured data.

We incorporate the above trend-detection mechanisms into Pathload and call it IMR-Pathload (*Interrupt Moderation Resilient Pathload*). We then evaluate it in Emulab and PlanetLab in the following section.

5 Performance Evaluation

5.1 Emulab Experiments

We investigate estimation accuracy of IMR-Pathload for different interrupt delays and report its relative estimation errors e_A in Table 3. As the table shows, IMR-Pathload produces available bandwidth estimates for all cases with 88−99% accuracy, which is significantly better than that of Pathload (see Table 2). Notice in the table that even with a large interrupt delay $\delta = 500$ μs, IMR-Pathload measures the paths within $e_A = 10\%$ error in all studied cases (recall that Pathload can measure none of the paths if $\delta > 125$ μs as discussed in §3.1).

5.2 Internet Experiments

In this section, we report experimental results obtained by measuring several Internet paths between Universities and a HP Lab in U.S. Measurement hosts

Table 4. Internet Experiment

Internet paths	Method	Available bandwidth estimates (Mb/s)				
		9 − 10 am	12 − 1 pm	3 − 4 pm	7 − 8 pm	11 − 12 pm
HP → Wustl	IMR-Pathload	12.2	11.9	13	12.8	13.1
	Pathload	−−	−−	−−	−−	−−
UMD → HP	IMR-Pathload	93	92.8	92.3	93.2	94.7
	Pathload	95.1	91.7	91.2	93.2	92.6
UMD → TAMU	IMR-Pathload	100	98.1	98.3	99.4	98.4
	Pathload	−−	−−	−−	−−	−−
HP → UMD	IMR-Pathload	12.9	11.8	13.3	12.3	12.6
	Pathload	20	−−	16.9	−−	−−

used in this study are located at HP (HP Labs), TAMU (Texas A&M University), UMD (University of Maryland), and Wustl (Washington University in St Louis). Note that we choose these paths simply for the convenience of accessibility. Also note that the purpose of these experiments is not to compare estimation accuracy of bandwidth estimators since we do not know exact characteristics of these paths. Instead, we use this example to assess how reliably IMR-Pathload measures Internet paths compared to Pathload.

For this purpose, we select 5 different periods of time in a day and run IMR-Pathload and Pathload three times for each time period to measure a particular path. When a tool produces bandwidth estimates reliably in all three times for each period, we report their average as its available bandwidth estimate. If the tool cannot estimate bandwidth at least once in three trials, we consider that the tool is not able to measure that particular path reliably in that period. For IMR-Pathload, we test both wavelet- and averaging-based algorithms, but report only wavelet-based estimates since the other produces similar results.

Table 4 shows bandwidth estimates produced by IMR-Pathload and Pathload. As the table shows, IMR-Pathload reliably produces available bandwidth estimates for all studied paths in all measurement time periods. Note that for a path (UMD → HP), Pathload also produces estimates that are similar to those of IMR-Pathload[4]. However, Pathload is unable to reliably measure the other three paths (HP → Wustl, UMD → TAMU, and HP → UMD).

6 Conclusion

This paper studied Pathload under a wide range of end-host interrupt delays and identified its estimation instability under non-negligible interrupt delays. We found that Pathload's instability stems from that its delay-trend detection mechanism is unreliable when probing packets are coalesced at the receiver. We overcame this problem using robust trend detection algorithms (called

[4] This agrees with the Emulab results, where Pathload shows accuracy that is similar to IMR-Pathload only if it is able to reliably measure the path (see Tables 2 and 3).

IMR-Pathload) and showed using Emulab and Internet experiments that IMR-Pathload greatly improves measurement stability of Pathload under various network settings.

References

1. C. Burrus, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice-Hall, 1998.
2. P. Craigmile, P. Guttorp, and D. Percival, "Trend Assessment in a Long Memory Dependence Model Using the Discrete Wavelet Transform," *Environmetrics*, vol. 15, no. 4, pp. 313–335, May 2004.
3. I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, Oct. 1988.
4. C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-Dispersion Techniques and a Capacity-Estimation Methodology," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 963–977, Dec. 2004.
5. Emulab. [Online]. Available: http://www.emulab.net/.
6. N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–974, Aug. 2003.
7. Interrupt Moderation Using Intel GbE Controllers. [Online]. Available: http://download.intel.com/design/network/applnots/ap450.pdf.
8. Iperf – The TCP/UDP Bandwidth Measurement Tool. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/.
9. M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth," in *Proc. Passive and Active Measurement Workshop*, Mar. 2002.
10. G. Jin and B. L. Tierney, "System Capability Effects on Algorithms for Network Bandwidth Measurement," in *Proc. ACM IMC*, Oct. 2003, pp. 27–38.
11. S. Kang, X. Liu, A. Bhati, and D. Loguinov, "On Estimating Tight-Link Bandwidth Characteristics over Multi-Hop Paths," in *Proc. IEEE ICDCS*, Jul. 2006.
12. S. Kang, X. Liu, M. Dai, and D. Loguinov, "Packet-Pair Bandwidth Estimation: Stochastic Analysis of a Single Congested Node," in *Proc. IEEE ICNP*, Oct. 2004, pp. 316–325.
13. R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 67–78.
14. B. Melander, M. Björkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *Proc. IEEE GLOBECOM*, Nov. 2000, pp. 415–420.
15. R. Prasad, M. Jain, and C. Dovrolis, "Effects of Interrupt Coalescence on Network Measurements," in *Proc. Passive and Active Measurement Workshop*, Apr. 2004.
16. V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Proc. Passive and Active Measurement Workshop*, Apr. 2003.
17. G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.