Non-Cooperative Available Bandwidth Estimation towards ADSL links

Daniele Croce, Taoufik En-Najjary, Guillaume Urvoy-Keller and Ernst W. Biersack

Institut Eurecom

Sophia-Antipolis, France

Email: {croce,ennajjar,urvoy,erbi}@eurecom.fr

Abstract—**Existing tools for the estimation of the end-to-end available bandwidth require control of both end hosts of the path and this significantly limits their usability. In this paper we present ABwProbe, a single-ended tool for available bandwidth estimation against non-cooperative hosts. Although ABwProbe is general enough to be used on any Internet path, we focus our attention on ADSL links exploring the possibility of measuring the downlink available bandwidth of a non-cooperative ADSL host. We study the effect of cross-traffic on the uplink, finding that only large packets may deteriorate ABwProbe's measurements and we present two techniques to detect and filter the effect of uplink cross-traffic.**

I. INTRODUCTION

The available bandwidth (avail-bw) of an end-to-end path is a fundamental metric for the operation of some applications and is directly coupled to the network load. Many tools have been developed for avail-bw estimation, including Spruce [1], Pathload, IGI/PTR, pathChirp and TOPP (see [2] for a complete survey of these tools). An interesting evaluation of some of these techniques can be found in [3].

Broadband technology, on its side, has become the normal way to access the Internet for over 221 million users [4], and this number should double by the next four years [5]. In many countries, such as Korea, Netherlands, Japan and recently US, over half of all households have a broadband access, with peaks of 90% penetration. These networks are vital for applications such as video-on-demand, online gaming, video streaming, P2P and content delivery systems, in essence for all the new and emerging uses of the Internet. Broadband access is paving the way for bandwidth intensive applications but represents still the fundamental bottleneck for achieving higher performances [6]. Over 62% of broadband lines are constituted by DSL networks.

Researchers have up to now devoted little attention to residential broadband networks, mostly because of the lack of means to analyze them without explicit cooperation from the residential hosts or the ISPs. Indeed, in order to measure the avail-bw, most existing tools need access to both end hosts of the path. This is impractical when the hosts belong to different organizations (usually the case) and severely limits the usability of these tools. In this paper we present ABwProbe, a new tool for estimating the avail-bw in non-cooperative environments, and we explore its use towards residential ADSL links. Our tool sends TCP ACKs as probes and expects from the other host to answer with the corresponding RSTs, similary to the approach used in [6] to measure several characteristics of ADSL hosts, (but not the available bandwidth). Focusing on ADSL, we show that asymmetry and encapsulations can invalidate the measurements if the ACK probes are not correctly sized. We study the impact of cross-traffic and propose two techniques to filter out cross-traffic effects. We evaluate the efficacy of these methods. Finally, we evaluate some techniques that significantly reduce the time needed to measure slow links such as ADSL.

II. BACKGROUND AND RELATED WORK

Pathneck [7] was designed to locate Internet bottlenecks. It uses the TTL field in the IP header to solicit ICMP probes from non-cooperative hosts and routers. Long trains of back-to-back packets are used to estimate the Asymptotic Dispersion Rate (ADR) which, as proven in [8], is in between the avail-bw of the path and the capacity. Pathneck thus measures something different than ABwProbe and additionally suffers from ICMP filtering and rate limiting.

In [6], various techniques are used to measure several characteristics of non-cooperative ADSL hosts. To measure the capacity, packets are sent at high rate for 10 seconds in order to completely saturate the link. In contrast, ABwProbe measures the avail-bw and in a much less intrusive way.

A. Available Bandwidth estimation

On a generic link i with capacity C_i, the avail-bw A_i is defined as the residual capacity $A_i = C_i \times [1 - u_i]$, where u_i is the average link utilization. Given a path, the link in the path with the minimum avail-bw is called the *tight link*, and its avail-bw is the end-to-end avail-bw A of the path. Since the avail-bw is a function of both the load and the physical capacity of the link, it is possible that the *tight link* is not the link with smallest capacity, called the *narrow link*.

Existing tools for measuring the avail-bw can be classified in two categories, depending on the model used, Probe Gap Model (PGM) tools and Probe Rate Model (PRM) tools [1]. Comparing the spacing between the probes at the sender and at the receiver, PGM tools estimate the load of the path from which an avail-bw measure is obtained. PRM tools are instead based on the principle of *self-induced congestion*: let A be the avail-bw of the path measured and suppose we send a sequence of packets at rate R; if the sending rate is lower than the avail-bw ($R < A$) the probes should be received at the destination at the same rate R at which they where sent. On the contrary, if the sending rate was higher than the avail-bw ($R > A$), the packets will queue and will be received with increasingly higher delay and at a lower rate. By iteratively sending probes at different rates and testing if the avail-bw is higher or lower, PRM tools converge to the avail-bw value A.

PGM tools require *a priori* knowledge of the capacity of the bottleneck link. Additionally, the PGM model has been criticized and proven inaccurate in [9]. For these reasons, ABwProbe is inspired by Pathload, a popular PRM tool, and we extend, adapt and improve the method especially for non-cooperative ADSL hosts.

B. Pathload

To better understand ABwProbe, we explain briefly Pathload's functioning (see [10] for detailed explanation). Pathload maintains an upper bound R^{max} and a lower bound R^{min} to converge to the avail-bw A using a "binary search" approach: starting from rate $R =$ADR, the algorithm tests if $R > A$ by sending 12 independent trains (or *fleets*) of 100 packets each and measuring the One-Way Delay (OWD) for each packet. If the sending rate is higher than the avail-bw, the packets will be spaced out (due to queuing) and the OWD series will show an *increasing trend*. The trend is computed independently for the 12 fleets. If at least 70% of the fleets detect an *increasing trend* (*no trend*), then R^{max} (R^{min}) is updated with the current value of R because

$R > A$ ($R < A$). The following iteration will be done at rate $R = (R^{max} - R^{min})/2$. The algorithm stops when the two bounds are closer than an estimation resolution $R^{max} - R^{min} \leq \omega$ and returns the two bounds as the avail-bw *variation range*.

Pathload uses two metrics to detect if a OWD trend is increasing or not: the Pairwise Comparison Test (PCT), that tracks the fraction of consecutive OWD pairs that are increasing, and the Pairwise Difference Test (PDT), that measures the overall OWD variation from the beginning to the end of the fleet. Let K be the number of packets per fleet. The OWD samples are divided in $\Gamma = \sqrt{K}$ groups and the median OWD D^k is taken from each group, with $k = 1, ..., \Gamma$. The metrics are defined as $PCT = \sum_{k=2}^{\Gamma} I(D^k > D^{k-1})/(\Gamma - 1)$ and $PDT = (D^\Gamma - D^1)/\sum_{k=2}^{\Gamma} |D^k - D^{k-1}|$ where $I(X)$ is one if X holds, zero otherwise.

III. ABwProbe: Non-Cooperative Estimation of the Available Bandwidth

In a non-cooperative environment, it is not possible to measure the OWD because there is no control on the receiver. So the idea is to use the RTT in place of the OWD. ABwProbe is able to measure the avail-bw between a source and a non-cooperative host by leveraging on the TCP protocol behavior: when a host receives an ACK for a connection not established, it replies with a TCP reset segment (RST). Hence, ABwProbe sends TCP ACKs to the non-cooperative host, evaluates RTT estimates from the corresponding RSTs and uses them instead of the OWDs of the forward path[1]. Ideally, if nothing interacts with the RST flowing back, the RTT will be higher than the OWDs but the statistical properties will be preserved so that the trends in the RTT series will perfectly reflect the OWD trends: if the RTT increases (*increasing trend*), this will be a sign that the fleet rate $R > A$ and the probes are congesting the buffer. If the RTT keeps stable (*no trend*), then $R < A$. In reality however, the RTT measures the characteristics of both the forward and the reverse paths. Thus, if the RSTs traverse a link with low avail-bw, the RTT measure will regard the reverse path and not the forward path. Additionally, cross-traffic on the reverse path can significantly alter the statistics carried by the RSTs. In Section III-A we will explain how to correctly measure the forward path, while in

---

[1]ABwProbe uses different sequence numbers in each probe in order to match the RSTs with the corresponding ACKs even in presence of loss or reordering. Note that other TCP packets with different flags can be used to solicit a RST, but the important thing is that the receiver replies to the probes.

Section III-B we propose two methods to filter RTT samples affected by cross-traffic.

Note that the RTT can also be used in some cases to measure the reverse path. Although ABwProbe is capable to measure both the forward and the reverse paths, due to space constraints in this paper we restrict the discussion only to the avail-bw estimation on the forward path.

A. Exploiting ADSL Asymmetry

By definition, the avail-bw of a path is the residual capacity of the *tight link*. The real question now is: which link of the path we are measuring is the tight link? Is it on the *forward path* traversed by the ACKs or on the *reverse path* where the RSTs flow? It is known that the core of the Internet has high capacity and only little of this capacity is used [7]. So, except some very particular cases, the *tight link* must be at the edge of the network. In the case of the ADSL, the uplink has the lowest capacity so it is surely the *narrow link* and most probably the *tight link* too, although not necessarily.

Since our objective is to measure the avail-bw on the forward path, *i.e.*, on the downlink of the ADSL, we must modify the ACK size so that the measurements are not affected by the lower capacity of the uplink. The idea is simple and is based on the observation that on the reverse path the RSTs will have 40 bytes length no matter how large the ACK that generated it is.
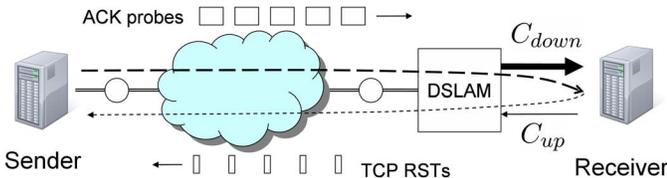
Fig. 1. **Non-cooperative estimation:** The measuring host sends TCP probes with the ACK flag on. Since no connection was previously open, the receiver will transmit TCP RSTs in reply. The ADSL link is characterized by its asymmetry as the downlink capacity C_{down} is greater than the uplink capacity C_{up}.

Figure 1 represents a typical scenario for our measurements. Let A_{down}, A_{up}, S_{ACK} and S_{RST} be the downlink avail-bw, uplink avail-bw, ACK size and RST size respectively. Let also $\gamma_L = L_{ACK}/L_{RST}$ be the ratio between the ACK's rate on the forward path (L_{ACK}) and the load generated by the corresponding RSTs on the uplink (L_{RST}). Since for every ACK there will be a RST flowing back, $\gamma_L = S_{ACK}/S_{RST}$. As $40 \leq S_{ACK} \leq$ MTU while $S_{RST} = 40$, then $1 \leq \gamma_L \leq$ MTU/40. We

analyze now separately the two cases when the uplink is the *tight link* or when the downlink is the *tight link*.

1) If the tight link is the uplink ($A_{up} < A_{down}$): Using pure ACKs of 40 bytes, the load generated on the forward and on the reverse path will be equal ($\gamma_L = 1$) so the tool would simply measure the lowest avail-bw of all the link traversed, certainly not A_{down} since A_{up} is lower. To measure the downlink, we must increase the load on the direct path and reduce it on the reverse path up to the point where $L_{RST} < A_{up}$ so that the RSTs are not constrained by the uplink. If the probes traversing the forward path are big, say $S_{ACK} = 1500$ bytes, then on the uplink the rate of the RSTs will be $\gamma_L = 1500/40 = 37.5$ times lower than the corresponding ACK load on the downlink, since $L_{RST} = L_{ACK}/\gamma_L$.

A major problem in computing γ_L precisely is the Layer 2 overhead: most ADSL configurations adopt ATM (together with encapsulation protocols like PPPoA or PPPoE) and this has great impact especially on small packets. While an MTU packet is normally fragmented in 32 cells (1696 B total, 13% overhead), a 40 bytes RST can inflate up to 106 bytes, or 2 ATM cells, which is 2.65 times higher. Thus, at Layer 2 the ratio between an MTU and a RST packet size can be reduced down to $\gamma_L = 32/2 = 16$. Since usually the L2 infrastructure is not exactly known, it is good practice to be conservative and allow some extra margin when setting the ACK size. However, varying the ACK size only might not be enough to overcome the great asymmetry between downlink and uplink[2].

2) If the tight link is the downlink ($A_{up} > A_{down}$): Since $\gamma_L \geq 1$ (RSTs are always 40 bytes and ACKs can not be smaller than 40 bytes), in this case the tool will always output the downlink avail-bw A_{down} independently of the value of γ_L.

In summary, by adjusting the probe size ratio γ_L appropriately, it is always possible to measure the downlink avail-bw A_{down}.

B. Filtering uplink cross-traffic

When measuring the downlink avail-bw, we would like the statistical information carried by the RSTs to be maintained throughout the reverse path. In this way, the RTT measured will reflect the OWDs of the probes on the forward path. When cross-traffic is present, noise

---

[2]By interleaving ACKs with other probes which do not generate replies (such as TCP RSTs) it is possible to increase the downlink load further while keeping the amount of RSTs on the uplink fixed: for example, if 30% of the probes do not generate a reply, then γ_L will increase by 30% and L_{RST} will reduce accordingly.

is introduced in the measurements and this can cause some overestimation because an increasing trend in the RTT samples can be "hidden" by this noise. Indeed, if the increasing trend is not computed correctly, the algorithm will not detect that $R > A$ and will erroneously increase R in search for an increasing trend, hence the overestimation.

Because of the great difference in capacity, the impact of the cross-traffic in the core Internet is negligible compared to the impact of cross-traffic on the ADSL uplink. With the typical low capacity of the uplink, the transmission of an MTU size packet can take up to several tens of milliseconds. This causes the RSTs to queue behind an MTU size packet for very long time, significantly altering the RTT values. A clear example of this can be seen in Figure 2, where the RTT suddenly increases when RSTs queue behind a large cross-traffic packet. Over 15 RSTs are buffered altogether and then sent back-to-back, thus destroying all the statistical information they carry with them. The PCT metric will be spoiled by the RTT going up and down many times and the PDT metric will judge the overall increase as not relevant compared to the RTT variations.
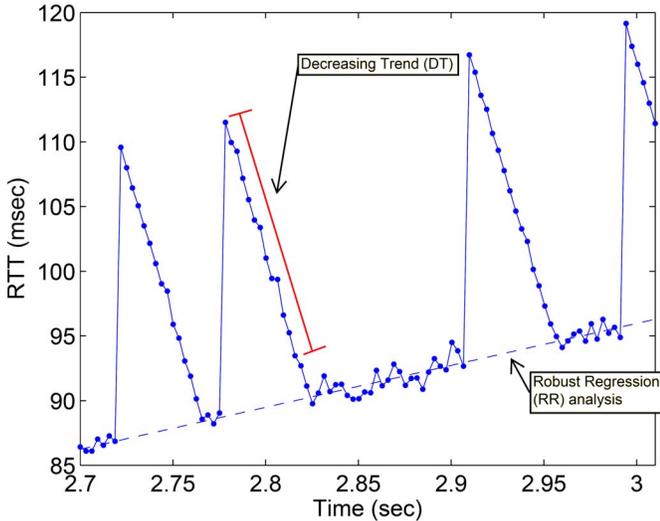
Fig. 2. **Cross-traffic on the uplink:** RTT evolution of a fleet with *increasing trend* in presence of cross-traffic constituted by MTU size packets. The trend is hidden by the impact of the large packets that compress a large fraction of samples.

Table I gives some quantitative examples of the impact of the uplink cross-traffic on the downlink measurement without load. Downlink capacity was 3.9 Mbps, uplink capacity was 578kbps, γ_L was set to 16 so that the bandwidth consumption of the RSTs flowing uplink was less than 250 kbps and the condition $L_{RST} < A_{up}$ was

respected (note that $L_{ACK} = L_{RST}/\gamma_L = 4$ Mbps). Table I shows that small cross-traffic packets have almost no impact on the measurements and that, when MTU packets are involved, even very low rates of cross-traffic can significantly bias the results towards overestimating the avail-bw.

	Uplink cross-traffic rate			
	none	100kbps	150kbps	200kbps
X-traffic	Downlink avail-bw variation range (Mbps)			
MTU pkts	**3.86-3.92**	4.01-4.38	4.37-4.95	4.58-5.99
40B pkts	**3.86-3.92**	4.04-4.10	4.00-4.12	4.02-4.09

TABLE I
IMPACT OF CROSS-TRAFFIC: SMALL PACKETS HAVE LITTLE IMPACT WHILE MTU ONES CAUSE OVERESTIMATION.

Mitigating the effect of uplink cross-traffic is not an easy task. Small packets usually do not impact the measurement, so the real objective for us is to remove the effect of big packets. Signs of cross-traffic on the uplink are two RSTs with an abnormal gap between each other followed by several back-to-back RSTs. Obviously, techniques used to combat cross-traffic must be robust enough to face more complicated and less clear scenarios than the one depicted in Figure 2: cross-traffic packets can be smaller than MTU and several packets can be sent in a small burst falling between two RSTs. We have developed two techniques to filter out the effect of uplink cross-traffic that have given promising results:

Detecting cross-traffic signatures (Decreasing Trend, DT): RSTs that have queued for a long time at the uplink produce an RTT sample that is significantly higher than others. As illustrated in Figure 2, the first RST behind the cross-traffic packet experiences the highest delay while the following packets queued in the buffer experience lower delay and the RTT measured gradually reduces until the impact of the cross-traffic vanishes. Afterwards the samples reflect the downlink measurement again. The RSTs that have been queued do not carry information on the avail-bw any more and thus their RTT samples should be discarded. In our tool, if the RTT exhibit a decreasing trend over 8 or more consecutive samples, these samples are discarded.

Robust Regression (RR): The DT technique just explained eliminates all RTT samples affected by large cross-traffic packets flowing on the uplink. Smaller cross-traffic packets however affect fewer samples and the decreasing trend that follows can easily be confused with the normal variability of the RTT. In these situations

the DT method is not effective, especially because of the (conservative) threshold of 8 consecutive decreasing samples. Considering that RTT variations in the probes are generally much smaller than the spikes provoked by uplink cross-traffic, we can filter out the bad samples affected by smaller cross-traffic packets with a robust statistical technique such as the Iteratively Re-weighted Least Squares (IRLS) method [11]. IRLS is a general algorithm for minimizing an objective function and is used in robust statistics to eliminate outliers. By iteratively applying the weighted least squares method, IRLS assigns a weight w between 0 and 1 to all samples and gradually reduces the weight of outliers while keeping other samples with $w \sim 1$. In our experiments we have found that good samples have always weight $w > 0.7$ while outliers have $w \sim 0$. In ABwProbe we discard all samples that have been marked with weight less than 0.1 which is conservative enough to avoid discarding good samples.

C. Reducing the running time

Most of the existing literature has focused on measuring the avail-bw over high speed links. Very little attention has been given to the measurement over slower links such DSL. A big issue over slow links is the time needed to provide an estimate since tools like Pathload may require up to 5 or even 10 minutes to provide a result. This is because the number of probes sent is generally constant but the transmission times are much higher. The parameters we can modify to reduce the estimation time are:

- the number of probing fleets N,
- the waiting time T between different fleets,
- the number of probes K sent in every fleet.

The number of fleets N and number of probes per fleet K must be sufficiently large to provide an accurate measurement and depend on the measurement strategy. The waiting time T is necessary so that fleets do not interfere with each other and the corresponding measurements are independent.

In ABwProbe we have significantly reduced the running time by carefully optimizing the first two points, T and N, the third one, K, being the least significative in terms of time. Pathload uses $N = 12$ in all circumstances, so during every iteration 12 fleets are sent. However, in the early phases when the probing rate R is far from converging to A, sending all these fleets is superfluous because the trends in the RTT will be striking as $R >> A$ or $R << A$. Fewer fleets are often sufficient for a good estimate and thus ABwProbe will stop sending

fleets as soon as a clear result is available. In particular our algorithm stops if one of the following conditions is satisfied:

1) the first fleet reveals a trend so pronounced that no more measurements are necessary to be confident in the result;
2) the first 3 fleets completely agree (100%);
3) out of 5 fleets, 4 agree (80%);
4) out of 9 fleets, 7 agree (78%);
5) the maximum number of fleets have been sent (we set this value to 12, as in Pathload).

Concerning the waiting time T between fleets, we choose a large waiting time when the last fleet has an increasing trend, but a much shorter time when there is no trend at all. An increasing trend is a sign that the probes have impacted the bottleneck buffer (and the existing flows) with a rate higher than the avail-bw and thus we must give time for the buffer to "recover" and drain all probes. This is not necessary if the fleet's rate was lower than the avail-bw and thus we can send another fleet right away. In ABwProbe we set T to be equal to the duration of a fleet whereas the default time in Pathload is 9 times this value.

These improvements are particularly important to speedup the early phases of the measurement while maintaining good accuracy: the running time for our tool is reduced up to 90% compared to Pathload as illustrated in Table II.

IV. EVALUATION

We have tested ABwProbe on some real ADSL hosts under our control carefully evaluating the accuracy of the tool and the effectiveness of the cross-traffic filtering. The measuring host was connected to the Internet with a 100Mbps Ethernet link with at least 90 Mbps avail-bw both directions so we are confident that the tight link was always the ADSL. The results showed here are relative to only one host for illustrative reasons, but results on all other hosts were similar.

First of all, we compare ABwProbe against Pathload without cross-traffic interfering on the uplink. Since traffic constituted by large packets is the most difficult to treat, we test the accuracy of the tool by loading the downlink at various data rates using large MTU packets. The capacity of the downlink was 3.9 Mbps while the capacity of the uplink was 578 kbps. As shown in Table II, our tool gives excellent results matching closely Pathload's. The time needed to correctly measure the avail-bw is significantly lower for ABwProbe in all cases. At low rates, Pathload sends only one fleet instead

of the default 12, thus reducing the running time instead of increasing it (the time to transmit the fleets increases but reduction of the number of fleets is predominant).

Load	none	1 Mbps	2 Mbps	3 Mbps
PL (Mbps)	3.86-3.92	2.96-3.02	1.97-2.09	1.01-1.07
Time (sec)	293.1	282.2	113.9	92.91
ABP (Mbps)	3.86-3.90	2.94-3.01	1.97-2.03	1.01-1.07
Time (sec)	33.3	41.0	48.7	61.5

TABLE II
ACCURACY AND RUNNING TIME: DOWNLINK AVAIL-BW MEASURED WITH PATHLOAD (PL) AND ABWPROBE (ABP).

In order to analyze the effect of cross-traffic on the uplink, we measure the downlink without any load so that the rate of RST on the uplink is maximized. We used $\gamma_L = 16$ thus almost 50% of the uplink capacity is consumed by the RSTs. We then inject MTU cross-traffic packets on the uplink being careful to leave enough avail-bw for the RSTs to go through (the condition $A_{up} > L_{RST}$ must be respected) and we compare the accuracy of ABwProbe with and without applying the filtering techniques illustrated in Section III-B. From Table III, it is clear that the impact of cross-traffic has almost disappeared using the DT filter, while the RR alone works properly if cross-traffic is not too high. This is because at higher loads of cross-traffic a large fraction of RSTs are affected, sometimes over 60%, and the RR will not discard so many outliers.

	Uplink cross-traffic rate			
	none	100kbps	150kbps	200kbps
	Downlink avail-bw variation range (Mbps)			
no filter	**3.86-3.92**	4.01-4.38	4.37-4.95	4.58-5.99
RR filter	**3.86-3.92**	3.82-4.06	4.02-4.87	4.55-5.90
DT filter	**3.86-3.92**	3.83-3.91	3.95-4.14	3.84-3.88

TABLE III
EFFECTIVENESS OF FILTERING TECHNIQUES: DECREASING TREND (DT) GIVES EXCELLENT RESULTS. ROBUST REGRESSION (RR) GIVES GOOD RESULTS IF CROSS-TRAFFIC IS LOW.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented ABwProbe, a single-ended tool for measuring the available bandwidth in non-cooperative environments. TCP ACKs are sent to solicit RSTs at the destination host and thus compute the RTT used for the estimation. We have examined the use of ABwProbe to measure the avail-bw of ADSL links, focusing our attention on the downlink. We have

showed that by tuning the ACK size appropriately it is always possible to measure the downlink avail-bw. We have analyzed the impact of cross-traffic and explained why large cross-traffic packets on the uplink can bias the measurement. We have also proposed two techniques, Decreasing Trend and Robust Regression, to filter out RTT samples affected by uplink cross-traffic and we have evaluated their effectiveness. Finally, we have significantly reduced the time to estimate the avail-bw compared to Pathload.

While we have presented ABwProbe as a tool to measure ADSL access links available bandwidth, its operational principles are general and should apply to any type of Internet path, asymmetrical or not. Future work includes both tests on different types of Internet paths and a more extensive validation of our tool on residential broadband networks.

REFERENCES

[1] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *IMC '03: Proc. of the 3rd ACM SIGCOMM conference on Internet measurement.* New York, NY, USA: ACM Press, 2003, pp. 39–44.

[2] R.-S. Prasad, M. Murray, C. Dovrolis, and K.-C. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, Nov. 2003.

[3] A. Shriram and J. Kaur, "Empirical evaluation of techniques for measuring available bandwidth," *IEEE INFOCOM 2007*, pp. 2162–2170, May 2007.

[4] OECD, "Oecd broadband statistics," June 2007. [Online]. Available: http://www.oecd.org/sti/ict/broadband

[5] L. Windsor Oaks Group, "Annual market outlook report," March 2006. [Online]. Available: http://www.broadbandtrends.com/Report%20Summary/2006/BBT%20GlobalBBOutlook2006%20061110%20TOC.pdf

[6] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proc. Internet Measurement Conference (IMC'07)*, Oct. 2007.

[7] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: algorithms, measurements, and implications," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM Press, 2004, pp. 41–54.

[8] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings of the IEEE Infocom*, Anchorage, Alaska, Apr. 2001.

[9] L. Lao, C. Dovrolis, and M. Y. Sanadidi, "The probe gap model can underestimate the available bandwidth of multihop paths," *Computer Comm. Review*, vol. 36, no. 5, pp. 29–34, 2006.

[10] C. Dovrolis and M. Jain, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *ACM SIGCOMM*, Pittsbugrh, USA, Aug. 2002.

[11] Å. Björck, *Numerical Methods for Least Squares Problems.* Philadelphia: SIAM, 1996.