ASSOLO: an Efficient Tool for Active End-to-end Available Bandwidth Estimation

Emanuele Goldoni
University of Pavia
Department of Electronics
27100 - Pavia, Italy
emanuele.goldoni@unipv.it

Giuseppe Rossi, Alberto Torelli
University of Pavia
Department of Computer Eng. and System Science
27100 - Pavia, Italy
giuseppe.rossi@unipv.it, alberto.torelli01@ateneopv.it

***Abstract*—End-to-end available bandwidth estimation is a crucial metric for bandwidth-dependent services such as multimedia streaming, peer-to-peer and gaming applications; it is also useful for quality of service verification and traffic engineering. This paper presents the details of ASSOLO, an efficient active probing tool for estimating the available bandwidth of a network path. The tool is based on the well-known concept of "self-induced congestion", and it features a new probing traffic profile called REACH (Reflected ExponentiAl Chirp) to test a wide range of possible rates with a single stream of packets. In addition, the program runs inside a real-time operating system and uses some de-noising techniques to improve the measurement process. Experimental results show that ASSOLO outperforms pathChirp, a state-of-the-art measurement tool, estimating the available bandwidth with greater accuracy and stability in presence of different cross-traffic sources. Moreover, we demonstrate that the use of a real-time operating system can increase the stability of the estimations lowering the impact of software context switches.**

***Keywords*-Available bandwidth, active network measurement, performance evaluation, real-time.**

I. Introduction

ASSOLO is a novel tool for available bandwidth estimation in packet-switched networks which has been originally introduced in [1]. This work extends some of the results presented in the original paper by investigating the performance of our tool in presence of poissonian cross-traffics. We also study the actual impact of a real-time operating system on the measurement process, and we provide more details on the filtering technique implemented into the program.

The *available bandwidth* of a network path is a crucial metric in quality-of-service management, traffic engineering or congestion control. Voice over IP (VoIP), peer-to-peer and video-streaming are examples of widely-used applications that could greatly benefit from the knowledge of the available bandwidth along an Internet path. For example, in [2] and [3] the importance of the available bandwidth is investigated respectively for peer-to-peer (P2P) networks and gaming-on-demand services. In [4] the authors focus instead on improving the perceived quality of video streaming through a dynamic path selection based on the measurement of network-layer metrics. Similarly, in [5] the authors propose a live broadcast platform where the video source is distributed to a number of clients organized in a peer-to-peer tree-structured overlay network. In this network the root node is also responsible for organizing and maintaining the position of each peer within the tree according to the available bandwidth and the latency between peers. The knowledge of the actual available bandwidth is also exploited in [6] to improve video streaming rate- and quality-adaptation decisions; results obtained through simulations show that an estimation algorithm can substantially increase streaming performance.

The same approach is also adopted in existing commercial products: Microsoft Windows Media Server includes a technology called Intelligent Streaming for on-demand and live media streaming over IP. This solution identifies the actual maximum throughput allowed by the network path using a end-to-end client/server system. This value is used to choose the best encoding rate which maximizes the quality of received media without overloading the network [7].

In principle, it would be possible to obtain estimates of the available bandwidth directly from intermediate routers along the network path; however, this is not feasible in practice due to technical and security reasons. Therefore, researchers have proposed several end-to-end measurement algorithms which infer the network characteristic transmitting a few packets and observing the effects of intermediate routers or links on these probe frames. Examples of probing tools which have emerged in recent years are IGI [8], Spruce [9], Pathload [10], TOPP [11], [12], pathChirp [13], FEAT [14] and BART [15]. They differ mainly in the structure of probe streams and in the algorithms used to estimate available bandwidth from the received packets. Nevertheless, producing reliable estimations in real-time still remains challenging: the measurement process should be efficient, accurate, non-intrusive and robust at the same time. Moreover, the algorithm should adaptively apply to different types of networks and cross-traffics, and must be able to produce fast periodic estimations in order to track bandwidth fluctuations. As a result, as noted in [16]–[18], current available bandwidth estimation techniques and tools are far from being ready to be applied in many applications and scenarios.

Compared to the tools mention above, our novel tool ASSOLO (Available-bandwidth Smart Sampling On-Line Tool) features a new probing traffic profile called REACH (Reflected ExponentiAl Chirp). A REACH tests a wide range of rates

and is more accurate in the center of the probed interval. Moreover, ASSOLO uses a combination of new and existing filtering techniques to improve the accuracy and stability of results. Finally, our tool runs inside a real-time operating system in order to minimize the impact of context switches on the measurement process.

The rest of the paper is organized as follows. In Section II we introduce the related work on available bandwidth estimation and we focus on the Probe Gap Model, the general measurement scheme adopted by our tool. Next, Section III illustrates the algorithm used to generate the REACH probing stream and the additional features introduced in our tool. An evaluation of ASSOLO is presented in Section IV, including results obtained comparing our solution to the state-of-the-art tool pathChirp both in terms of intrusiveness and accuracy. Finally, in Section V we conclude and we outline future works.

II. Related Work

Techniques for end-to-end available bandwidth estimation can be divided into two categories: active probing and passive measurements. The latter infer the required information from existing data transmissions while active probing techniques produce an estimation injecting dedicated probe traffic into the network.

Passive measurements do not require dedicated packets to perform the estimation: useful information is obtained from traffic originated by active connections providing a particular service. In this context, the idea of using TCP for network measurements has attracted a lot of studies: RTT values [19] or ACK arrival times [20], [21] have been used on the sender's side to infer the available bandwidth from existing transmissions. These methods are lightweight and fast but they can be applied only to network paths that have recently carried traffic. Moreover, congestion control algorithms, buffers and competing connections may influence the achievable throughput of a single TCP connection, thus altering the accuracy of estimations [22].

Active measurement techniques use probe packets to measure the end-to-end delays introduced by existing cross-traffic (Figure 1). These methods require instrumentation at both ends of the path; moreover, the probe traffic injected into the network may affect the performance of other applications and actually alter the available bandwidth. In addition, some tools require a long measurement time and use hundred of packets before producing an estimation. The majority of existing tools belong to the Probe Gap Model (PGM) or the Probe Rate Model (PRM).

In the Probe Gap Model, a tool sends a single probing pair or train; it exploits then the dispersion of packets on the receiver side to calculate the available bandwidth. The main assumption of this model is that the link with the minimum available bandwidth is also the link having the minimum capacity. This is probably the biggest limit of this approach: the hypothesis is not valid for many Internet paths and can results in significant underestimations of the available

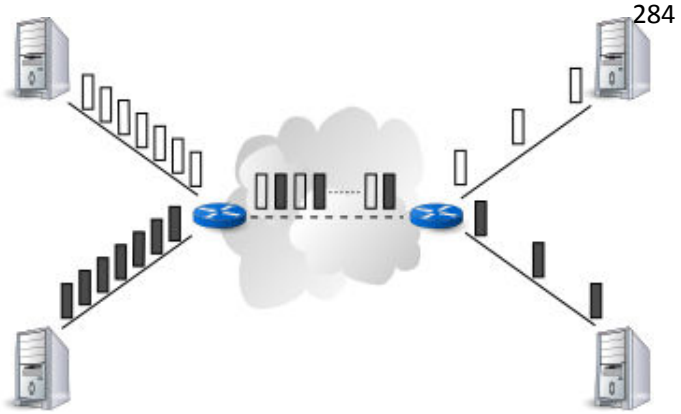

Fig. 1. The spacing effect on multiple traffics over a congested network path.

bandwidth over multi-hop links [23]. Notable tools based on the Probe Gap Model are Spruce [9], IGI [8] and Delphi [24].

Delphi [24] assumes a multi-fractal model for the cross-traffic. The main idea in this tool is that the spacing of two probing packets at the receiver can provide an estimate of the amount of traffic at a link. Spruce [9] is based too on direct probing and it uses tens of packet pairs to collect available bandwidth estimations. The input rate of pairs is chosen to be roughly around to the capacity of the path, which is assumed to be known. Moreover, packets are spaced with exponential intervals to emulate a poissonian sampling process. IGI [8] uses a sequence of about 60 unevenly space packets to probe the network and the gap between two consecutive packets is increased until the average output and initial gaps match.

The Probe Rate Model, instead, is based on the concept of self-induced congestion. The underlying idea is quite simple: if a sequence of packets is sent at a rate lower than the available bandwidth along the network path, then the arrival rate of packets at the receiver will not exhibit any notable variation and it will match with the sender's rate. On the other hand, if the sending rate exceeds available bandwidth, one or more intermediate queues will fill up and the probe traffic will experience delays. Thus, the measurement is performed through the research of the turning point at which the probe stream starts seeing an increasing trend. The PRM model has proved to be accurate and it is used in many estimation tools, such as TOPP, Pathload, pathChirp, FEAT and BART.

TOPP [11], [12] and Pathload [10] use a constant bit-rate stream, sending pairs or trains of packets at a given rate and changing this rate every round. TOPP increases linearly the sending rate in successive streams, trying to find out the exact turning point. Pathload on the other hand varies the probing rate using a binary search scheme and the final output, result of multiple measurements, is a variation range rather than a single estimate. Since multiple trains are required to produce a single estimation, the intrusiveness of these techniques is quite high and the measurement process is time-consuming.

PTR [8] is an active probing algorithm which sends several probing packets to detect background traffic. The method com-

pares the time interval at the source with that of destination and then uses the timings to estimate the value of available bandwidth.

pathChirp [13] sends a variable bit-rate stream called *chirp*, which consists of exponentially spaced packets. A chirp allows to probe the network path over a wide range of rates injecting only one stream – if the delays show an increasing trend starting from a particular packet, the associated rate is used to infer the unused capacity. pathChirp can estimate available bandwidth sending only one chirp: this feature makes the measurement process fast and lightweight. However, pathChirp samples the lower rates more frequently than the higher rates. Therefore the tool is less accurate if actual available bandwidth is not located nearby the beginning of the probing range. Smoothed-chirp (S-chirp) is a similar approach based on iterative probing and originally proposed in [25].

BART [15] relies on sequences of packet pairs sent at randomized rates. This tool uses also a Kalman filter to track the evolution of available bandwidth in real-time and to filter out noisy observations. BART is lightweight, efficient and non-intrusive; however, the tool is still in development and it is not freely available. MR-BART is a extension of the original BART method which employs multi-rate probe packet sequences to achieve faster convergence and more accurate estimations.

FEAT [14] is a recent tool which features a probe pattern called *fisheye stream*. A fisheye stream consists of packets of equal size which are sent at a changing rate, from a lower bound to a maximum probing rate. The tool identifies also an interval, called "focus region", where the available-bandwidth is most likely to be. Inside this region the sampling frequency is higher and the number of packets sent for each sampling rate is larger. This approach creates a more identifiable turning point but it also makes the measurement process intrusive.

While BART and FEAT look quite promising, it is difficult to compare them to other state-of-the-art tools: the results presented by the respective authors have been obtained only through simulations or using specific Internet paths, and to our best knowledge the two programs have never been released publicly.

III. ASSOLO

ASSOLO is an available bandwidth estimation tool which has been originally presented in [1]. Unique to this tool is a new probe traffic profile called REACH (Reflected ExponentiAl Chirp), which tests a wide range of rates using a single stream of packets and injecting a negligible amount of traffic into the network. The tool introduces also some techniques to minimize the impact of different sources of errors on the estimation process.

A. Probing stream

ASSOLO is based on the concept of "self-induced congestion" – it tests different rates using a single stream of packets, and then infers the available bandwidth harnessing the information about the relative delays. This approach has a twofold advantage: it requires neither clock synchronization nor clock-offset knowledge between the two end-hosts probing the network. However, it is important to consider that the first packet of the train itself does not have any associated rate. Instead, it is used as a reference value to calculate all successive *relative* queuing delays within a stream.

The novel REACH probing traffic profile tests multiple rates with a single stream, and it is more accurate at the center of the stream, where the actual available bandwidth is likely to be. A similar idea was originally proposed in [14], but our method introduces a different spacing algorithm and sends less packets. Compared to pathChirp, the stream used by ASSOLO is different too – both tools use a sequence of packets with increasing delays, but the shape of the traffic and the delays within a stream are not the same.

The REACH stream used by our tool tests different rates increasing the instantaneous packet rates from a lower bound L to a maximum rate U. The first k packets of the stream probe values lower than the center $H = \frac{U+L}{2}$; additional k packets test values between H and the maximum probing rate U. However, the probing rates do not increase linearly in a REACH. Instead, the density of the stream increases as well as values approach the center of the interval $[L, U]$. Then, once the rate H has been tested, the probing density start decreasing. The same can be said for the accuracy of the estimation, since it is proportional to the density of the probing stream.

The maximum relative accuracy of ASSOLO's estimations is defined by the parameter σ. Given the probing range, the absolute error S around the center of the probing interval is calculated as:

$$S = \sigma \left(\frac{U - L}{2} \right). \tag{1}$$

Moreover, the algorithm uses a coefficient γ to control how fast the density of streams changes. This parameter reminds the *spread factor* used by pathChirp, although the two resulting trains are quite different. ASSOLO uses by default $\sigma = 5\%$ and it sets γ to 1.2. However, it is important to note that the choice of these parameters is arbitrary – values should be assigned according to the specific requirements of the target application. Decreasing γ and σ, the tool would send more packets but it should result in a more accurate estimation; similarly, increasing these value should reduce both intrusiveness and accuracy.

An additional parameter Δ_x is also needed to better describe the REACH stream generated by ASSOLO. The function of this auxiliary coefficient is to describe the gap between two consecutive packets of the stream, and it is defined as follows:

$$\Delta_x = S \cdot \gamma^{|x-1|} \tag{2}$$

and combining Equations 1 and 2 we get:

$$\Delta_x = \sigma \frac{U - L}{2} \gamma^{|x-1|} \tag{3}$$

Starting from the center H of the probing interval towards the upper bound U, instantaneous packet rates in a REACH are

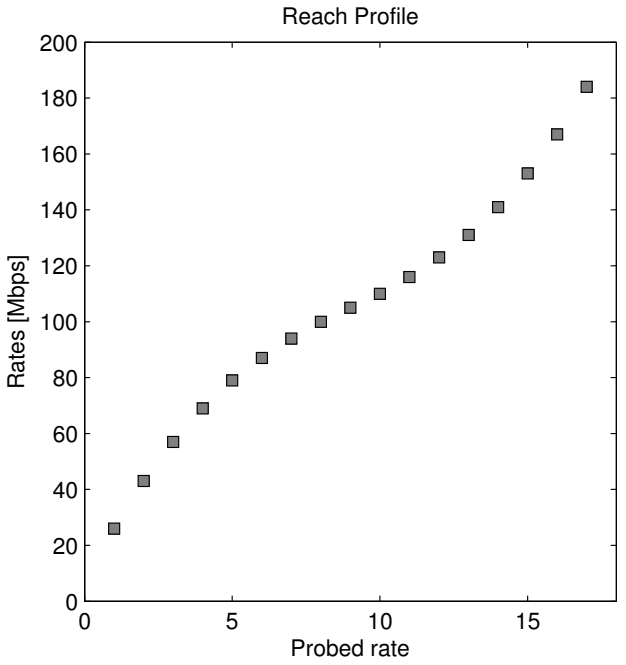

Fig. 2. Distribution of packets in a REACH stream.

$H, H+\Delta_1, H+\Delta_1+\Delta_2, H+\Delta_1+\Delta_2+\Delta_3, ...$ A more formal description of instantaneous probing rates R_x tested by this stream is:

$$\begin{cases} R_x = H, & \text{if } x = 1 \\ R_x = R_{x-1} + \Delta_x, & \forall x > 1, R_x < U \end{cases} \quad (4)$$

On the other hand, probing rates from the center towards the lower bound are $H, H-\Delta_1, H-\Delta_1-\Delta_2, H-\Delta_1-\Delta_2-\Delta_3, ...$ The instantaneous rates tested by a REACH can then be described as:

$$\begin{cases} R_y = H, & \text{if } y = 1 \\ R_y = R_{y-1} - \Delta_y, & \forall y > 1, R_y > L \end{cases} \quad (5)$$

The resulting stream is shown in Figure 2. As also the name REACH (Reflected ExponentiAl CHirp) suggests, the profile is symmetric: the right and the left part look like two mirrored exponential functions.

Since the function is symmetric, we can analyze only the right part of the stream – the same considerations would also apply to the left one. ASSOLO uses k packets to test values between H and the upper bound U. Thus, the instantaneous rate R_k associated with the k^{th} packet is the maximum probing rate. We can write this condition as:

$$U = R_k = H + \sum_{i=1}^{k} \Delta_i \rightarrow U - H = \sum_{i=1}^{k} \Delta_i \quad (6)$$

If we substitute the values of Δ_i and H, we get:

$$U - H = \sigma \frac{U-L}{2} \sum_{i=1}^{k} \gamma^{|i-1|} \quad (7)$$

$$U - \frac{U+L}{2} = \sigma \frac{U-L}{2} \sum_{i=1}^{k} \gamma^{|i-1|} \quad (8)$$

$$\frac{U - \frac{U+L}{2}}{\sigma \frac{U-L}{2}} = \sum_{i=1}^{k} \gamma^{|i-1|} \quad (9)$$

$$\frac{\frac{U-L}{2}}{\sigma \frac{U-L}{2}} = \sum_{i=1}^{k} \gamma^{|i-1|} \quad (10)$$

$$\frac{1}{\sigma} = \sum_{i=1}^{k} \gamma^{|i-1|} \quad (11)$$

In addition, the value of the truncated sum is:

$$\sum_{i=1}^{k} \gamma^{|i-1|} = \frac{\gamma^{k+1} - 1}{\gamma - 1} \quad (12)$$

Combining Equations 11 and 12, we get:

$$\gamma^{k+1} = \frac{\gamma - 1}{\sigma} + 1 \quad (13)$$

which leads to

$$k = log_\gamma \left(\frac{\gamma - 1}{\sigma} + 1 \right) - 1 \quad (14)$$

Actually we should define R_k as the maximum sending rate *not exceeding* the upper bound U. Equation 6 should then take the form:

$$U \geq H + \sum_{i=1}^{k} \Delta_i \quad (15)$$

Hence the correct value of k is:

$$k = \left\lfloor log_\gamma \left(\frac{\gamma - 1}{\sigma} + 1 \right) - 1 \right\rfloor \quad (16)$$

As we mentioned before, a REACH uses the first k packets of the stream to probe values lower than the center $H = \frac{U+L}{2}$. Then, the stream probes the rate H; finally, other k packets tests values between H and the maximum probing rate U. As a result, a REACH probes $2k+1$ rates exploiting relative delays between probe packets. Therefore, our tool needs to send an additional packet at the beginning of the REACH. The total number N of packets used by ASSOLO to probe $2k+1$ rates is:

$$N = 1 + (2k+1) = 2k+2 \quad (17)$$

Since we know the size of a REACH, we can also combine Equations 4 and 5 and describe the rates probed by a REACH profile as:

$$R_j = H + \text{sign}\left(j - \frac{N}{2}\right) \cdot S \cdot \frac{\gamma^{|j - \frac{N}{2}|} - 1}{\gamma - 1} \quad (18)$$

B. End-hosts predictability

A fundamental difficulty with the existing measurement tools stems from a number of issues on both end-hosts and network paths [26]: system timing, hardware errors and end-to-end pathologies could produce a considerable amount of noise in the individual network observations. For example, the Linux kernel is a time sharing operating system designed to give a fair share of the CPU in a multi-user environment [27] – even some kernel services like memory allocation and system calls exhibit some non-deterministic timing behavior.

Network measurement tools have strict operational deadlines between the arrive of a packet and the application's response to that event – the same can be said for the sender side, where the packet sent by the application should ideally start with no delay. In [28] the impact of context switching on the measurement process is analyzed in depth. Some tests conducted in our lab confirmed that a significant amount of noisy observations is due to the non-deterministic behavior of the operating systems hosting the sender and the receiver. To accommodate deadlines on both the end-hosts we decided to use a real-time operating systems (RTOS), which can guarantee predictability and accurate system timings for applications. ASSOLO runs inside a GNU/Linux system with RT-Preempt [29], [30] patch enabled, thus using a fully preemptible kernel with high-resolution kernel timers. In order to minimize the impact of context switches on the bandwidth estimation process, the tool gets the highest priority on both the end-host systems while probing the network.

Our program could be easily ported to other real-time operating systems, since it is written in C language and uses standard system calls. However we decided to use the RT-preempt approach, which makes the software much more portable and easier to deploy and maintain over a large network infrastructure.

Compared to other Linux real-time approaches, such as RTAI [31] and Xenomai [32], RT-Preempt is not a hard real-time approach in strict sense: processes can incur a latency that is not deterministic and no guarantees are usually provided on the feasibility of a given task set. Although this real-time extension to the Linux kernel suffers from the above-mentioned limitations, it greatly improves the performances of many applications and the responsiveness of the whole system, thus providing adequate service for most applications that need real-time determinism [33]. Moreover, no special programming libraries are required: the applications compiled for RT-Preempt Linux can be also used on a standard, non real-time Linux system with negligible adaptations.

C. Observations filtering

Like the end-hosts, also intermediate routers can be heavily affected by predictability issues: interrupt coalescence, clock resolution and context-switching delays are all factors that can potentially modify timings of the probe traffic, therefore introducing errors. Moreover, almost all existing tools assume the hypothesis of fluid cross-traffic [34], ignoring the discrete nature of packets. However this non-deterministic behavior of intermediate nodes depends on the specific network path and it can not be easily controlled or even described. As a result, most of existing available bandwidth estimation techniques produce noisy observations [35], [36].

A vast majority of available bandwidth estimation tools introduce filtering techniques: for example, Moving Average, Exponential Weighted Moving Average (EWMA), Wavelets or Kalman filters have been successfully adopted in [13], [15], [37]–[40] to attenuate noise and local random fluctuations, converting noisy values into a reliable estimate.

The idea of using such a solution in this context is based on the predictability and long-term stability of the Internet. Typically, the available bandwidth of an Internet path shows strong correlation and a certain degree of stability over intervals that span from several minutes to a few hours [14], [41]. Given a new observation, an effective filtering technique can produce a new estimate of the available bandwidth combining both the most recent observation and the old values.

For example, the Exponentially Weighted Moving Average (EWMA) filter uses one or more observed values O_k and outputs a new estimation E_i calculated as follows:

$$E_i = \alpha E_{i-1} + (1-\alpha)O_i. \tag{19}$$

This filter is used by some estimation tool like Abing [37] and Yaz [38]. However, the difficulty with the EWMA technique lies in the choice of the exponential weight α. With large values of α, the old estimates are given more importance and the filter is slow but stable; agility is instead achieved by keeping α small. Ideally, the filter should be adaptive, setting the value of α according to the current circumstances: sharp and non-persistent changes can at first be treated as noise using lower weights α_i. However, if the change persists, the filter should quickly converge to the new value. Equation (19) should then take the form:

$$E_i = \alpha_i E_{i-1} + (1-\alpha_i)O_i. \tag{20}$$

Lowpass EMA [42], *Stability* [43] and *Error Based* Filters [43] are three existing techniques designed around this philosophy. Although they have been proposed a couple of years ago, to our best knowledge none of them has actively been employed in an available bandwidth estimation tool.

In [44] we originally proposed the use of Vertical Horizontal Filter (VHF) in such a context. The VHF filter is a modified EWMA technique borrowed from the financial world [45] which can dynamically modify its behavior according to trends identified in the temporal evolution of available bandwidth according to the same principles of the three above mentioned filters. The dynamically exponential weight α_i in (20) is computed as:

$$\alpha_i = \beta \frac{\Delta_{max}}{\sum_{t=i-M}^{i} |O_t - O_{t-1}|} \tag{21}$$

where Δ_{max} is the gap between the maximum and the minimum values in the M most recent observations. We set β as $\frac{1}{3}$ and the window size $M = 10$, although these parameters

were obtained empirically and a careful choice could bring further improvements.

We performed a series of simulations to investigate the effectiveness of different filtering techniques on the available bandwidth estimation process. Compared to the methods mentioned above, we found that the VHF filter leads to better results in many cases and shows greater stability. Our experiments also indicated that there is no need to fine tune the VHF filter every time some network conditions change. A detailed description of VHF and a comparison between different linear filtering techniques can be found in [44].

Results persuaded us to employ the Vertical Horizontal Filter, which is used inside our tool to cope with noisy observations and to estimate the actual available bandwidth from raw measurements.

D. Excursions segmentation

According to the basic principle of PRM's self-induced congestion, an instantaneous sending rate higher than the actual available bandwidth results in increasing queuing delays at receiver; otherwise, packets sent by a tool will experience no delays. This model is valid also for tools which probe multiple rates with a single train, like ASSOLO does – the last instantaneous probing rate which does not result in an increasing queuing delay is considered a simple estimate of available bandwidth. However, this approach oversimplifies reality, lacking, for example, to consider cross-traffic bursty behavior and end-host interrupt coalescence effects.

Traditional network adapters generate an interrupt for each received frame, thus generating up to thousands of internal signals per second in high-speed networks. These interrupts consume a lot of system's resources and introduce a significant amount of context switches, resulting in a CPU overhead. [46]

To mitigate the effects of this issue, some network adapters recently introduced the support for Interrupt Coalescence (IC) [47]. This solution decreases the processing overhead buffering multiple packets before generating a single interrupt for the burst of frames. A similar approach has been introduced in NAPI [48], a modification to the device driver packet processing framework of Linux kernel. NAPI mixes interrupts with a polling approach to implement an adaptive interrupt coalescing which modifies its behavior according to the actual network load. This solution usually results in improved performances for high-speed networking. Although IC decreases the per-packet processing overhead, it introduces also non-deterministic queueing delays, thus altering the time spacing of packets in a probing train. As noted in [49], IC can be detrimental to TCP self-clocking making the traffic more bursty, and it has a negative effect on the accuracy of active and passive bandwidth measurements.

The typical profile of queuing delays in a train is often non-monotonic. For example, Figure 3 shows the typical queuing delays of a chirp sent by pathChirp: one or more excursions produced by bursts return to zero, while a final excursion ends with increasing queuing delays.

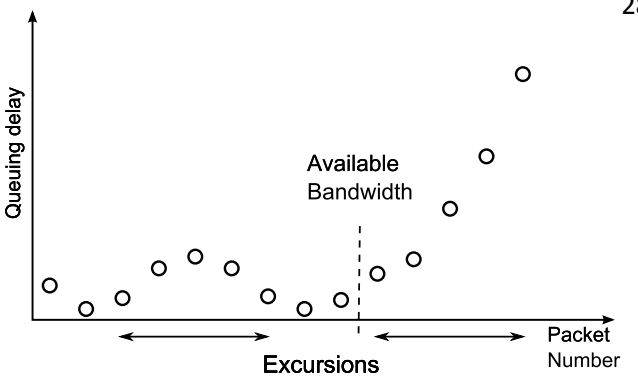

Fig. 3. Typical queuing delays in a chirp.

function EXCURSION(q, i, F, L)
  $j \leftarrow i + 1$
  $q_{max} \leftarrow 0$
  while $(j \leq N)\, AND\, (q_j - q_i > q_max/F)$ **do**
    j++ ▷ Count excursion's packets
  end while
  if $j \geq N$ **then**
    return j ▷ Non-ending Excursion
  end if
  if $j - 1 \geq L$ **then**
    return j ▷ Excursion
  else
    return i ▷ Not an excursion
  end if
end function

Fig. 4. Pseudo-code for the pathChirp's excursion segmentation algorithm [13].

The authors of pathChirp introduced a smart segmentation algorithm to cope with this kind of burstiness effects, detecting increasing delays belonging to a cross-traffic bursty transient. The main goal of pathChirp's excursion segmentation algorithm is to identify potential starting and ending packet i and j respectively for an excursion. Potentially, every packet i where queueing delay q_i starts increasing could be a starting point of an excursion. We define the end of the excursion as the point where the queuing delay returns to zero or where it has decreased by a factor F from the maximum queueing delay experienced during this interval. Moreover, if the distance between these two packet is long enough, for example longer than a threshold L, then all packets between i and j form an excursion. On the other hand, the last excursion identifies the congested region and it does not terminate. The pseudo-code of the procedure is presented in Figure 4 while a detailed description of the whole algorithm can be found in the original paper of pathChirp [13]. Since this solution proved to be quite effective to cope with burstiness, ASSOLO adopts exactly the same technique to analyze the queuing delays of each single REACH and to identify the correct turning point.

E. Availability

Additional implementation details and a copy of the source code of ASSOLO are all freely available at *http://netlab-mn.unipv.it/assolo/* or through the authors. Future developments and data reports will be published at the same location.

IV. Results

In order to evaluate our estimation method, the performance of ASSOLO has been studied in a controlled testbed environment. In addition, we compared the intrusiveness and the accuracy of our solution with pathChirp, a similar state-of-the-art measurement tool, in presence of poissonian or constant bit rate (CBR) cross-traffics.

The testbed configuration is shown in Figure 5. Two computers using Ubuntu GNU/Linux are connected together through a Fast Ethernet cross-cable and serve as routers. Two other machines of the testbed simulate a source of controlled traffic flows using the D-ITG tool [50], which loads the network generating synthetic flows of known properties and statistical distributions. Finally, the sender and the receiver for each measurement tool use additional PCs running Ubuntu GNU/Linux with a standard or real-time kernel. Prasad et al. in [51] showed that each store-and-forward device introduces an additional serialization latency in a packet's delay. This can result in a consistent underestimation of the hop's capacity. Therefore, we provisioned the network with two Fast Ethernet switches in order to introduce an additional potential source of errors during tests.

The topology of the testbed is quite simple but sufficient to evaluate the performance of a measurement tool: for example, the same configuration has been used in [52] and [17] to perform an experimental comparison of different available bandwidth estimation tools.

We adopted the default configurations for both probing tools: ASSOLO uses $\sigma = 5\%$ and γ to 1.2 while the γ of pathChirp has been initially set to 1.2. Since results obtained in [13] showed that pathChirp generally performs better with larger packets, the packet size for both tools was 1000 byte. Finally, the upper and the lower bandwidth bounds U and L were respectively equal to 200 and 10 Mbps; however, both tools automatically adjust the values if the range is too narrow. A complete list of all the configuration parameters of testbed's devices and tools is provided in [53].

A. Intrusiveness

The intrusiveness of pathChirp and ASSOLO can be easily compared. From [36] we know that a *chirp* is composed of N packets, where N can be calculated as follows:

$$N_{chirp} = \left\lfloor 2 + \frac{1}{log\gamma} log\left(\frac{U}{L}\right) \right\rfloor .$$

The size of the stream sent by pathChirp depends on the upper (U) and lower (L) rate bounds. However, pathChirp automatically reduces or increases the probing range if it is too wide or too narrow: as a result, the tool sends on average 15-20 packets.

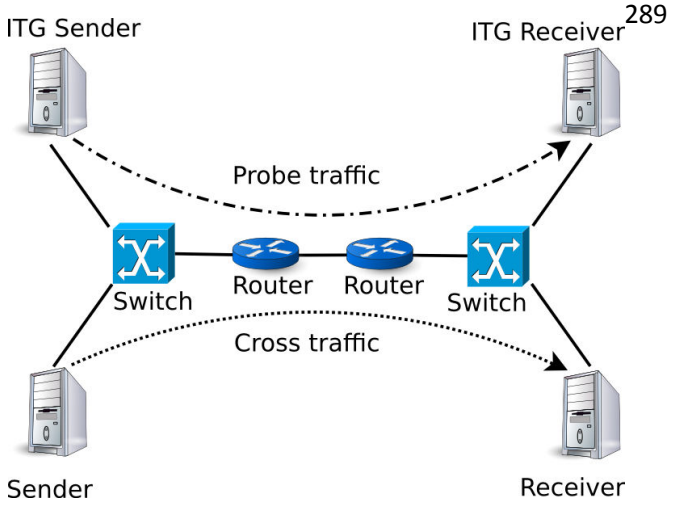

Fig. 5. Testbed configuration.

On the other hand, the length of a REACH only depends on the two parameter σ and γ. In section III-A we calculated the size of a REACH probe as:

$$N_{reach} = 2 \cdot \left\lfloor log_\gamma \left(\frac{\gamma - 1}{\sigma} + 1\right) - 1 \right\rfloor + 2.$$

Hence, our algorithm send always 18 packets using default values of σ and γ.

Our experiments show that the amount of traffic injected by ASSOLO and pathChirp is comparable and extremely limited. Using the default parameters, the measurement process of both tools takes less than one second to produce an estimation over links with a capacity higher than 1 Mbps. However, the two methods are based on the concept of self-induced congestion, i.e., the estimation is performed by injecting probe traffic at a rate higher than the available bandwidth of the network path. The drawback of this approach is that the bottleneck node is congested by the probe traffic – the existing cross-traffic is delayed, and its packets' timings can be significantly affected by the measurement process.

B. Accuracy

We tested both pathChirp and ASSOLO in the presence of different sources of cross-traffic with varying intensity. We generated CBR cross-traffic of 64, 32 and 16 Mbps and, finally, we turned off the traffic source. We evaluated both tools in each cross-traffic scenario, repeating the measurement process 10 times for each algorithm: averaged results are shown in Figure 6. Then, we repeated the same tests simulating different sources of poissonian cross-traffic with increasing average traffic load. The results obtained after 10 runs are shown in Figure 7.

Our experiments show that pathChirp constantly overestimates available bandwidth and measurements are quite unstable. This is a well-know problem of pathChirp: similar results have been obtained in [15], [17], [54]. On the other hand, we found that 80% of ASSOLO's estimations exhibit a relative error lower than 15%. Figure 8 shows an example of

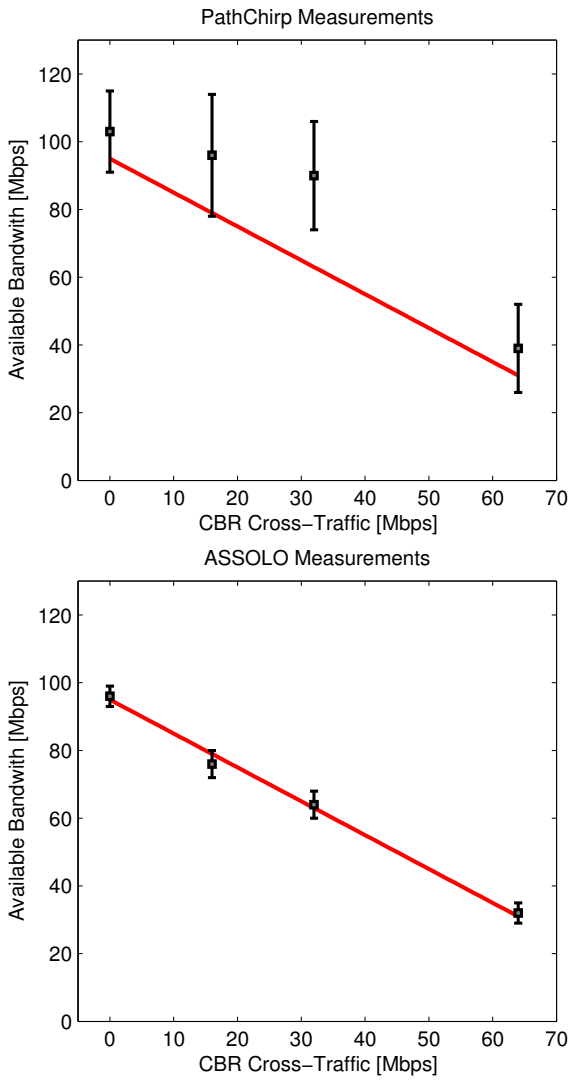

Fig. 6. Measurements obtained in presence of different CBR cross-traffics

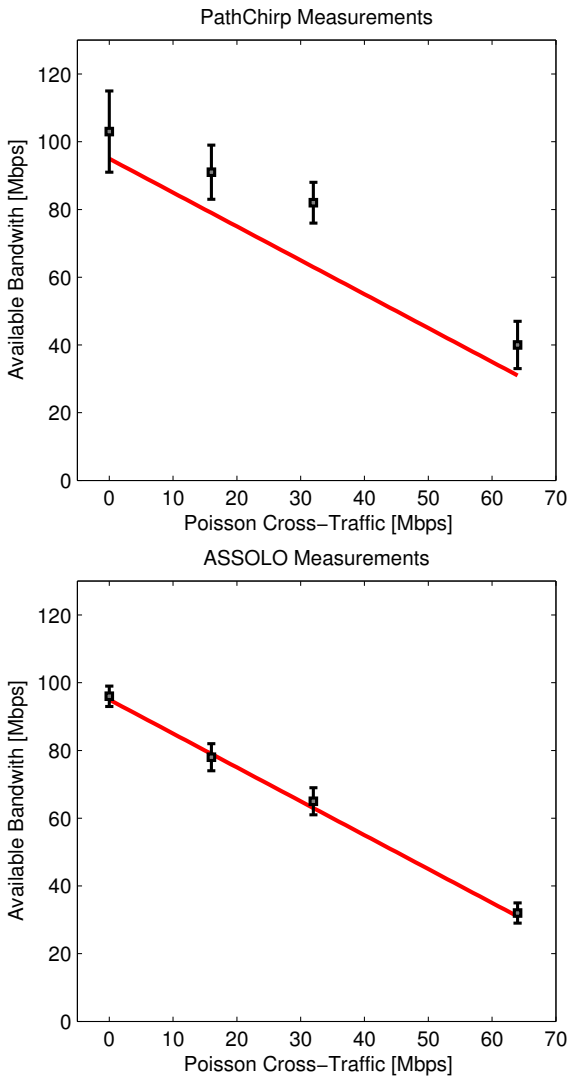

Fig. 7. Measurements obtained in presence of different poissonian cross-traffics

measurement performed in our testbed while the network path is loaded with a Constant Bit Rate cross-traffic of 32 Mbps: the difference between the two tools is notable both in terms of accuracy and stability.

It is worthy of remark that the accuracy of the two tools does not seem to depend on the nature of the cross-traffic – the performances are almost identical using either a CBR source or poissonian distributed packets.

C. Stability

We have analyzed the impact of a real-time operating system on the ASSOLO's measurement process. We performed a few tests with the real-time feature enabled and then we disabled it before repeating the estimation procedure with our tool. A sample comparison of the measurements obtained in the two cases is shown in Figure 9: the average value is correct in both configurations but the real-time feature provides much more stability. Although more investigations would be required, preliminary results confirm that the use of a real-time environment can effectively reduce the impact of different non-deterministic sources of error.

The same experiments could also be repeated for a longer observation interval, in order to catch possible long-term oscillations or biases in the estimations obtained with a non real-time system.

V. Conclusion and Future Work

In this work we presented the details of ASSOLO, an active probing tool which features an efficient measurement scheme for end-to-end available bandwidth estimation in packet-switched networks. Moreover, we described some de-noising techniques and detailed the real-time operating system used by our tool to improve the estimation process.

Preliminary experiments revealed that our algorithm is non-intrusive and accurate, estimating the available bandwidth with greater accuracy and stability with respect to the pathChirp measurement tool developed by the Rice university.

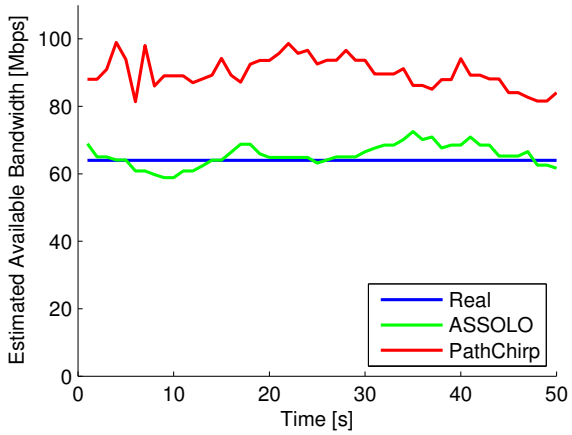

Fig. 8. An example of estimation using pathChirp and ASSOLO.

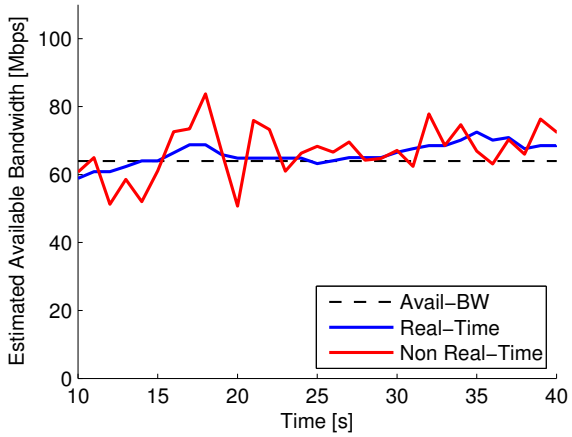

Fig. 9. Measurements using either a real-time operating system or not.

The testbed we used is quite simple and the synthetic cross-traffic does not fully catch the complexity of actual communication flows. We plan to test intensively the performance of our tool over actual Internet paths and in presence of realistic cross-traffic traces. We will also include a study of the actual accuracy, intrusiveness and robustness when dynamic traffic patterns are presents. An extensive comparison of our approach with other state-of-the-art tools is needed too. Above all, BART and FEAT are recent tools which seem to perform better than the original pathChirp: a comparative study will be conducted as soon as the code of these software will be freely available.

Since the bounds of ASSOLO's probing interval have to be set manually at start up, a coarse estimation of the current available bandwidth is required prior using our tool. We plan to introduce an initial self-configuring feature as proposed in [55], thus avoiding the need for any prior knowledge of the network path.

ACKNOWLEDGMENT

We would like to thank Dr. Davide Cavalca for giving the paper a critical reading and for providing us several helpful comments. We acknowledge also Dr. Alberto Savioli and Marco Schivi for their help during the setup of the laboratory testbed and the analysis of experimental results.

REFERENCES

[1] E. Goldoni, G. Rossi, and A. Torelli, "Assolo, a new method for available bandwidth estimation," in *Proc. IARIA International Conference on Internet Monitoring and Protection (ICIMP 2009)*, May 2009, pp. 130–136.

[2] C. Wu, B. Li, and S. Zhao, "Characterizing peer-to-peer streaming flows," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1612–1626, December 2007.

[3] J. P. Laulajainen, T. Sutinen, and S. Jarvinen, "Experiments with QOS-aware gaming-on-demand service," in *IEEE International Conference on Advanced Information Networking and Applications (AINA 2006)*, vol. 1, Apr. 2006, pp. 805–810.

[4] M. Jain and C. Dovrolis, "Path selection using available bandwidth estimation in overlay-based video streaming," *Computer Networks*, vol. 52, no. 12, pp. 2411–2418, August 2008.

[5] M. Favalli, L.and Folli, A. Lombardo, D. Reforgiato, and G. Schembra, "A bandwidth-aware p2p platform for the transmission of multipoint multiple description video streams," in *Proc. Italian Networking Workshop Reti.it 2009*, Jan. 2009.

[6] T. Tunali and K. Anar, "Adaptive available bandwidth estimation for internet video streaming," *Signal Processing: Image Communication*, vol. 21, no. 3, pp. 217–234, March 2006.

[7] M. Topic, *Streaming Media Demystified*. New York, NY: McGraw-Hill Professional, 2002.

[8] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, August 2003.

[9] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, pp. 39–44, October 2003.

[10] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. Passive and Active Measurement Conference (PAM 2002)*, Mar. 2002, pp. 14–25.

[11] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proc. IEEE Global Communications Conference (GLOBECOM 2000)*, Nov. 2000, pp. 415–420.

[12] A. Johnsson, B. Melander, and M. Björkman, "Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method," in *Proc. Swedish National Computer Networking Workshop (SNCNW 2004)*, Nov. 2004.

[13] 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 Conference (PAM 2003)*, Apr. 2003.

[14] Q. Wang and L. Cheng, "FEAT: Improving accuracy in end-to-end available bandwidth measurement," in *Proc. IEEE Global Communications Conference (GLOBECOM 2006)*, Nov. 2006, pp. 1–4.

[15] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J.-E. Mangs, B. Melander, and M. Bjorkman, "Real-time measurement of end-to-end available bandwidth using kalman filtering," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, Apr. 2006, pp. 73–84.

[16] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," in *Proc. ACM SIGCOMM Conference on Internet Measurement (IMC'04)*. Oct. 2004, pp. 272–277.

[17] A. A. Ali, F. Michaut, and F. Lepage, "End-to-end available bandwidth measurement tools : A comparative evaluation of performances," in *Proc. International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe 2006)*, Feb. 2006.

[18] C. D. Guerrero and M. A. Labrador, "On the applicability of available bandwidth estimation techniques and tools," in *Computer Communications*, vol. 33, no. 1, pp. 11–22, January 2010.

[19] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: new techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*. Aug. 1994, pp. 24–35.

[20] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "TCP westwood: congestion window control using bandwidth estimation," in *Proc. IEEE Global Communications Conference (GLOBECOM 2001)*, Nov. 2001, vol. 3, pp. 1698–1702.

[21] C. L. T. Man, G. Hasegawa, and M. Murata, "A new available bandwidth measurement technique for service overlay networks," in *Proc. IEEE/IFIP International Conference on Management of Multimedia Networks and Services (MMNS 2003)*, Sep. 2003, pp. 436–448.

[22] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 537–549, August 2003.

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

[24] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk, "Multifractal cross-traffic estimation," in *Proc. ITC Specialist Seminar on IP Traffic Measurement*, Sep. 2000.

[25] A. Pasztor, "Accurate active measurement in the internet and its applications," Ph.D. dissertation, Department of Electrical and Electronic Engineering, The University of Melbourne, 2003.

[26] H. Zhou, Y. Wang, X. Wang, and X. Huai, "Difficulties in estimating available bandwidth," in *Proc. IEEE International Conference on Communications (ICC'06)*, Jun. 2006, pp. 704–709

[27] The Linux Kernel. [Online]. Available: http://www.kernel.org

[28] Y. Ozturk and M. Kulkarni, "Dichirp: direct injection bandwidth estimation," *International Journal of Network Management*, vol. 18, no. 5, pp. 377–394, September 2008.

[29] Gnu/Linux Real-Time. [Online] Available: http://rt.wiki.kernel.org/

[30] K. Koolwal, "Myths and realities of real-time linux software systems," in *Proc. Real-Time Linux Workshop (RTLWS 2009)*, Oct. 2009. [Online]. Available: http://lwn.net/images/conf/rtlws11/papers/proc/p20.pdf

[31] RTAI: Realtime application interface for linux. [Online] Available: http://www.rtai.org/

[32] Xenomai: Real-time framework for Linux. [Online] Available: http://www.xenomai.org

[33] K. Yaghmour, J. Masters, G. Ben-Yossef, and P. Gerum, *Building Embedded Linux Systems*. Sebastopol, CA: O'Reilly & Associates, 2008.

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

[35] C. D. Guerrero and M. A. Labrador, "Experimental and analytical evaluation of available bandwidth estimation tools," in *Proc. IEEE Conference on Local Computer Networks (LCN 2006)*, Nov. 2006, pp. 710–717.

[36] E. Goldoni, "Nuovi approcci nella stima della banda disponibile in una rete a pacchetto," Master thesis, University of Pavia, 2007.

[37] J. Navratil and R. L. Cottrell, "Abwe: A practical approach to available bandwidth," in *Proc. Passive and Active Measurement Conference (PAM 2003)*, Apr. 2003.

[38] J. Sommers, P. Barford, and W. Willinger, "A proposed framework for calibration of available bandwidth estimation tools," in *Proc. IEEE Symposium on Computers and Communications (ISCC'06)*, Jun. 2006, pp. 709–718.

[39] S.-R. Kang and D. Loguinov, "IMR-Pathload: Robust available bandwidth estimation under end-host interrupt delay," in *Proc. Passive and Active Measurement Conference (PAM 2008)*, Apr. 2008, pp. 172–181.

[40] G. Urvoy-Keller, T. En-Najjary, and A. Sorniotti, "Operational comparison of available bandwidth estimation tools," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 39–42, January 2008.

[41] Y. Zhang and N. Duffield, "On the constancy of Internet path properties," in *Proc. ACM SIGCOMM Workshop on Internet Measurement (IMW'01)*, Nov. 2001, pp. 197-211.

[42] L. Burgstahler and M. Neubauer, "New modifications of the exponential moving average algorithm for bandwidth estimation," in *Proc. ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, July 2002.

[43] M. Kim and B. Noble, "Mobile network estimation," in *Proc. International conference on Mobile Computing and networking (MobiCom'01)*, Jul. 2001, pp. 298–309.

[44] E. Goldoni, G. F. Rossi, and P. Gamba, "Improving available bandwidth estimation using averaging filtering techniques," University of Pavia, Tech. Rep., 2008. [Online]. Available: netlab-mn.unipv.it/publications/tr-netlab2008-01.pdf

[45] A. White, "The vertical horizontal filter," *Futures Magazine*, vol. 20, no. 10, pp. 1–10, 1991.

[46] J. C. Mogul and K. K. Ramakrishnan, "Eliminating receive livelock in an interrupt-driven kernel," *ACM Transactions on Computer Systems*, vol. 15, no. 3, pp. 217–252, August 1997.

[47] Intel. (2003) Interrupt moderation using Intel gigabit ethernet controllers. [Online]. Available: http://download.intel.com/design/network/applnots/ap450.pdf

[48] J. H. Salim, R. Olsson, and A. Kuznetsov, "Beyond softnet," in *Proc. USENIX Annual Linux Showcase & Conference (ALC'01)*. Nov. 2001, pp. 165-172.

[49] R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in *Proc. Passive and Active Measurement Conference (PAM 2004)*, Apr. 2004, pp. 247–256.

[50] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, "D-ITG distributed internet traffic generator." in *QEST*. IEEE Computer Society, 2004, pp. 316–317.
A. Botta, A. Dainotti, A. Pescapè, "Multi-protocol and multi-platform traffic generation and measurement," in *IEEE Conference on Computer Communications (INFOCOM 2007), Demo Session*, May 2007.

[51] R. S. Prasad, C. Dovrolis, and B. A. Mah, "The effect of layer-2 store-and-forward devices on per-hop capacity estimation," in *IEEE Conference on Computer Communications (INFOCOM 2003)*, Mar. 2003, vol. 3, pp. 2090–2100.

[52] L. Angrisani, S. D'Antonio, M. Esposito, and M. Vadursi, "Techniques for available bandwidth measurement in ip networks: a performance comparison," *Computer Networks*, vol. 50, no. 3, pp. 332–349, February 2006.

[53] A. Torelli, "Sviluppo di una tecnica innovativa per la stima della banda disponibile," Master thesis, University of Pavia, 2008.

[54] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, and K. C. M. Fomenkov, "Comparison of public end-to-end bandwidth estimation tools on high-speed links," in *Proc. Passive and Active Measurement Conference (PAM 2005)*, Mar. 2005, pp. 306–320

[55] W. Tan, M. Zhanikeev, and Y. Tanaka, "Abshoot: A reliable and efficient scheme for end-to-end available bandwidth measurement," in *Proc. IEEE Region 10 Conference (TENCON 2006)*, Nov. 2006.