The Quest for Bandwidth Estimation Techniques for large-scale Distributed Systems

The Quest for Bandwidth Estimation Techniques for Large-Scale Distributed Systems*

Daniele Croce[†⋆], Marco Mellia[†] and Emilio Leonardi[†]
[†] Politecnico di Torino, Turin, Italy
[⋆] Université de Nice-Sophia Antipolis, Sophia Antipolis, France
Email: {firstname.lastname}@polito.it

ABSTRACT

In recent years the research community has developed many techniques to estimate the end-to-end available bandwidth of an Internet path. This important metric has been proposed for use in several distributed systems and, more recently, has even been considered to improve the congestion control mechanism of TCP. Thus, it has been suggested that some existing estimation techniques could be used for this purpose. However, existing tools were *not* designed for large-scale deployments and were mostly validated in controlled settings, considering only one measurement running at a time. In this paper, we argue that current tools, while offering good estimates when used alone, might not work in large-scale systems where several estimations severely interfere with each other. We analyze the properties of the measurement paradigms employed today and discuss their functioning, study their overhead and analyze their interference. Our testbed results show that current techniques are insufficient as they are. Finally, we will discuss and propose some principles that should be taken into account for including available bandwidth measurements in large-scale distributed systems.

1. INTRODUCTION

The end-to-end available bandwidth (avail-bw) is one of the most important characteristics of an Internet path. This metric is fundamental for the operation of many emerging applications, such as video streaming, online gaming, peer-to-peer and content delivery systems. Thus, the attention of the research community has focused in recent years on the problem of the avail-bw estimation, and several techniques and inference methods have been proposed in literature [1,3, 5,8,14,17], so that both accurate and fast estimation of the end-to-end avail-bw can today be obtained. More recently, the attention of the research community has then moved on how to exploit the avail-bw knowledge, so that it has been proposed for use in several algorithms: for i) server selection [7], ii) route selection [10], iii) admission control [2], iv) congestion control (in TCP as well) [11], and v) for peer-to-peer systems optimization [19].

The performance of avail-bw estimations techniques has been discussed in several articles where the tools have been tested separately in different settings and traffic conditions (see [15] and citations therein). However, existing studies analyze the performance of each tool in a controlled setting, where one single path is probed by only one pair of hosts at a time. In a more realistic scenario, instead, where a certain technique is employed by many hosts in the Internet, several nodes want to estimate the path properties simultaneously. Consider for example a P2P system, in which several thousands of peers periodically probe each other to optimize the content distribution. It is therefore not clear if the techniques proposed are suitable to be employed in large-scale distributed systems as well, and, if not, which modifications are required. The question we will try to answer is: *what if several hosts measure simultaneously?* Or, alternatively, would current avail-bw estimation tools interfere with each other in large-scale deployments? Finally, notice that the problem is particularly important on "critical" links, such as peering links between ASs or links connecting aggregation nodes (e.g., DSLAM or BRAS nodes) to the backbone, where a large number of hosts may perform measurements. To the best of our knowledge, this is the first paper that clearly points out this problem in the context of the avail-bw estimation.

The contributions of this article are the following: we analyze the properties of three popular avail-bw tools, namely Spruce, Pathload and pathChirp, which represent the prominent measurement methodologies presented in literature. We discuss their functioning, study their impact and analyze the interference caused by running several measurements at the same time. Our results show that all current techniques suffer from severe interference, some of them providing rather inaccurate results. Finally, we discuss some possible modifications to include avail-bw measurement techniques in large-scale distributed systems.

2. INTERFERENCE AND OVERHEAD IN CURRENT ESTIMATION TOOLS

Several bandwidth estimation tools have been proposed in the literature. They can be coarsely split into two classes, based on the probing techniques: *Probe Gap Model - PGM*,

*This work is supported by the "NAPA-WINE" Project (FP7-ICT-214412) of the European Union.

and *Probe Rate Model - PRM*. In the PGM, the avail-bw is directly inferred by observing the inter-packet-gap (IPG) measured on a packet pair injected by the sender. By observing the IPG "dispersion" of several probes, an estimation of the avail-bw is obtained. In the PRM, the sender iteratively injects trains of packets at different rates, allowing the receiver to understand if the path avail-bw is exceeded or not, e.g., by observing increased One-Way-Delay (OWD), or RTT. Table 1 reports a summary of some popular tools and the methodology they adopt.

Tool	Algorithm	Methodology	Inference Metric
IGI [8]	Iterative	PGM	Dispersion
Spruce [17]	Direct	PGM	Dispersion
Pathload [5]	Iterative	PRM	One-way Delay
pathChirp [14]	Direct[1]	PRM	One-way Delay
BFind [1]	Iterative	PRM	RTT

Table 1: Available Bandwidth Estimation tools.

In this paper, we limit the investigation to Spruce, Pathload and pathChirp, which represent the classic measurement methodologies today in use. Several other tools have been proposed in the literature, and the majority of them rely either on the Probe Gap or the Probe Rate model. Thus, we expect that results gathered using the three previous mentioned tools are indicative of PGM and PRM tools.

2.1 Spruce

Overview: Spruce estimates the avail-bw using *direct probing*. A packet pair is sent at the capacity rate C of the bottleneck link, i.e. with and input gap $\Delta_{in} = S/C$, being S the size of each packet. The output gap Δ_{out} between the packets is then measured at the receiver. The avail-bw is computed as

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

which has been proved to be an exact model at least in the case of single-hop scenarios. The technique used in Spruce is a classic example of Probe Gap Model paradigm because the relationship between the input and output gap of the probes is used explicitly to estimate the avail-bw. The PGM model requires the capacity C of the bottleneck to be known and assumes there is only one bottleneck on the path. Although the PGM model has been formally proved inaccurate in some multi-hop scenarios [12], other studies have shown that in practice Spruce performs similar to other tools [15–18]. Before providing the avail-bw estimate, Spruce repeats the measurement 100 times, with measurements emulating Poisson sampling, and the result is averaged among all samples.

Interference analysis: In a scenario where several measuring hosts are running Spruce at the same time, there are chances that two or more packet pairs will overlap as shown in Fig. 1. In this case, one of the two packets will be transmitted in between the other pair(s) and the initial probing gap Δ_{in} will be inflated to $k \times \Delta_{in}$, where k is the number of overlapping pairs. Note that already with *two* pairs overlapping, an output gap of $\Delta_{out} = 2 \times \Delta_{in}$ will lead in Eq. 1 to an avail-bw estimate of zero, i.e., an error of 100% on a single probe!

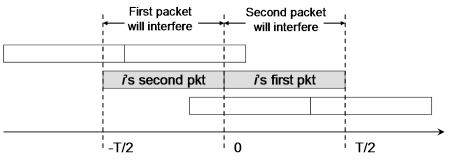


Figure 1: Interference between packet pairs.

When more Spruce senders interfere (or when the bottleneck link is overloaded), the Δ_{out} increases further, leading to a negative value of the avail-bw. Unfortunately, negative values cannot be blindly ignored because they represent transient periods of time in which the bottleneck queue fills up, which happens periodically in normal operating conditions. Therefore, the idea of distinguishing and filtering affected measurements by detecting negative estimates is not viable.

Overhead: Regarding the measurement overhead, the overall probing rate is set to be the minimum between 240 kbps and 5% of the bottleneck capacity C. While this overhead seems acceptable for a single running instance of Spruce, it becomes intolerable already with a handful of measuring hosts, with the double risk of (i) quickly congesting the bottleneck link and (ii) obtaining useless avail-bw estimates, because of the high probability of pairs interfearing with each other. Additionally, reducing the probing rate can reduce the total overhead, at the cost of waiting longer times to get the final estimation.

2.2 Pathload

Overview: Pathload is a classic example of PRM tool. The basic idea is to use the *self-induced congestion* principle. Let A be the avail-bw of the path. The sender inject a train of packets at rate R. If R is lower than the avail-bw ($R < A$,) then the probes will reach the receiver without any change. On the contrary, if the probing rate is too high ($R > A$), packets will queue at the bottleneck link and the probes will be received with increasingly high delay, and lower rate. By iteratively changing R, Pathload converges at the avail-bw value A performing a "binary-search" and updating a minimum and maximum bound: when it is detected that $R < A$, the R^{min} bound is updated to R (respectively $R^{max} = R$ if $R > A$) and at the next iteration $R = (R^{min} + R^{max})/2$. This way, Pathload captures the avail-bw minimum and maximum ranges, converging to A. The initial values of the bounds are $R^{min} = 0$ and $R^{max} =$ ADR[2]. The algorithm stops once a predefined measurement resolution is achieved (by default 4% of the ADR).

Interference analysis: Predicting the bias caused by several hosts running Pathload simultaneously is a challenging

[1]In [9], pathChirp has been classified as "iterative" because it does not use Eq. 1 to compute the avail-bw, and uses the OWD instead. However, since each packet train provides an independent avail-bw estimate without need of successive iterations, we prefer classifying the tool as "direct" instead.

[2]The Average Dispersion Rate (ADR) is the rate at which a burst of back-to-back packets is received, and has been proved to be an upper bound of the avail-bw [6].

task for several reasons. Depending on the interaction between the measurements, Pathload can take erroneous decisions at various stages of the binary-search process, with different consequences: an error in the final stages of the search, when the R^{min} and R^{max} bounds are already close to each other, has little impact on the final results; conversely, a wrong decision at the beginning of the process leads to a large measurement error. Notice that Pathload repeats the measurement 12 times, and then it takes the decision if $R > A$ or not only if the results agree in 70% of the cases. Finally, each probe is composed of 100 equally spaced packets, thus several competing measurements will share the avail-bw A in different proportions depending on the actual overlap, if complete or only partial. Also, the idle time between each probe at the same rate and between probes at different rate is automatically computed, so that predicting the overall interference is very complex. In our experimental evaluation, however, we will show that already few hosts can interfere destructively, heavily biasing the final results.

Overhead: Like other PRM tools, Pathload probes tend to consume *all* the available bandwidth, as the probing rate R converges to the avail-bw A. Therefore, the measurement overhead is very high. When the bandwidth is low, the pausing time between different probes is computed to reduce the average overhead to approximately 10% of avail-bw. However, when more measurements coexists, the bottleneck bandwidth may be easily consumed by parallel runs. Note that, compared to Spruce, the overhead introduced by Pathload is proportional to the avail-bw instead of the capacity. This has two important advantages: first, the additional traffic is guaranteed to consume part of the avail-bw without damaging on-going connections; second, the overhead automatically "scales" with the avail-bw and is accordingly reduced when the avail-bw is low.

2.3 PathChirp

Overview: PathChirp uses particular packet trains, called *chirps*, which consist of N packet-pairs sent with inter-packet-gap that is exponentially reduced. Each consecutive "packet-pair" probes the path with increasing rate. PathChirp thus probes for a wide range of rates sending a single train, from a lower rate L (the first two packets sent) up to an upper rate $U = \gamma^{(N-1)} \times L$. For fixed N, the *spread factor* γ controls both the granularity of the estimates and the range of rates probed for. The receiver detects which is the rate R_l of the last packet pair l received with the expected IPG after which the rate of the pairs is higher than the avail-bw A. Indeed, packets after pairs after pair l queue up at the bottleneck, so that additional delay is observed at the receiver. Fig. 2 shows an example of the queuing delay seen by the receiver. In more details, a delay increase (or *excursion*) is a symptoms of self-induced congestion and the path avail-bw is computed as a weighted average of the "per-pair" avail-bw A_i, defined as

$$A_i = \begin{cases} R_i & \text{if delay is increasing AND transient excursion} \\ R_l & \text{otherwise} \end{cases}$$

where R_i is the rate at which pair i is received. With "transient" excursions, we refer to all excursions that terminate before the end of the chirp and, in general, before rate R_l is reached – which represents the beginning of the final excursion. A more detailed description of the avail-bw estimation process can be found in [14].

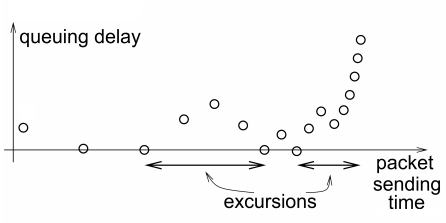


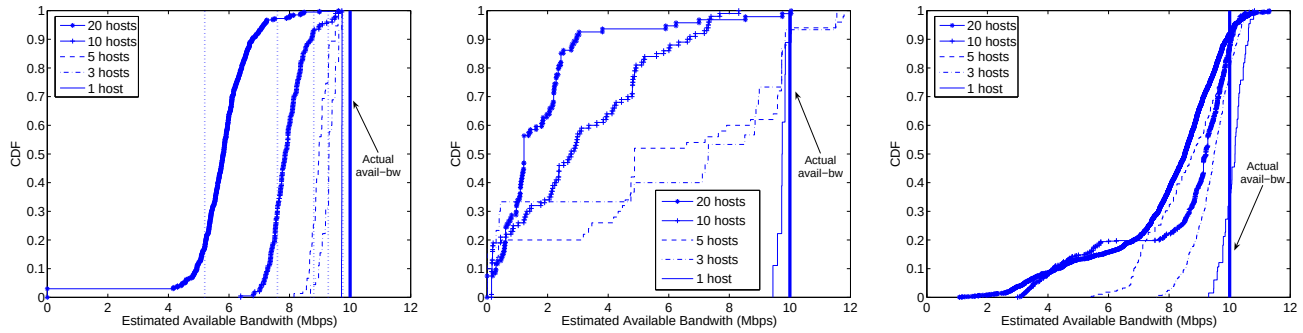
Figure 2: Queuing delay of a chirp (from [14]).

Interference analysis: PathChirp uses a sophisticated algorithm to compute the avail-bw. It takes into account both the last rate R_l above which it exceeds the avail-bw A and also the evolution of the queuing delay across the rest of the packet chirp before this point. Therefore, when another chirp interferes, either (i) it can generate a transient excursion on the other chirp, if the interfering chirp ends early enough, or (ii) the excursion caused by the interfering chirp overlaps with the last excursion and the avail-bw rate is reached in advance – the rate R_l is detected lower. In both cases, the avail-bw will be underestimated. An additional complication comes from the fact that pathChirp averages the results among several chirps (11 by default). Finally, desynchronization among different pathChirp instances causes variable overlapping periods, thus, it becomes almost impossible to predict the interference introduced by competing measurements. Our results show that in some conditions the results become very inaccurate because of competing chirps.

Overhead: pathChirp has many tunable parameters, from the spread factor γ, to the upper and lower rates U and L, from the average probing rate R (300 kbps by default), to the number of chirps to average over. Additionally, pathChirp automatically tunes the U and L bounds so that the avail-bw A is well in between, i.e. $L < A < U$. Therefore, even when only one host is running pathChirp, part of the chirp will consume *more* then the actual path avail-bw. This creates spikes of bursty traffic which can affect other traffic. While nodes could absorb the impact of one chirp, several overlapping chirps quickly fill the bottleneck buffer so that severe losses may occur. Reducing the average probing rate R increases the pausing time between chirps but also makes the estimation process slower and does not reduce the total amount of probing traffic nor the impact of the chirps on the queues.

3. EXPERIMENTAL EVALUATION

In this section we compare quantitatively the performance of the different techniques when multiple hosts are simultaneously estimating the avail-bw. We used a dedicated testbed composed of a total of 62 identical machines equipped with 100 Mbps network cards and connected through Ethernet switches. All hosts run the same version of Linux. Half of the hosts acted as senders and the other half as receivers. For experiments that required traffic load, two additional hosts where used to inject and drain UDP traffic. We created a single bottleneck topology, with one host routing all



(a) Spruce (b) Pathload (c) pathChirp

Figure 3: Interference between measurement tools. Existing techniques clearly underestimate the avail-bw.

the traffic between the senders and the receivers, and we forced the router NIC to 10 Mbps. This topology is similar to the ones used in [11, 14] and it is the simplest and less problematic topology possible. Therefore, the performance will further deteriorate in the real Internet, where cross-traffic on non-bottleneck links, or multiple bottlenecks are typically present. We emphasize that our goal is to observe the mutual interference, rather then the accuracy of each single tool.

To better control the number of overlapping measurements, we ran the tools continuously on the probing hosts. In order to randomize the interference between measurements and avoid synchronization, the starting instant was set according to an exponentially distributed time, with average chosen to be half of the measuring time of each tool. We then let the experiments run for several minutes. An increasing number of active hosts N where present in each experiment. Each tool parameters were left to their default values. Fig. 3 compares the performance of Spruce, Pathload[3] and pathChirp with different values of N. In this first set of experiments, no background traffic was present so that $A = C = 10$ Mbps. From the figures it is clear that

- when $N = 1$, all estimations are very close to the actual avail-bw, proving that in this simple scenario, they provide good results.
- when N increases, *all* tools suffer from mutual interference and they can significantly underestimate the avail-bw.
- Spruce, which has a simple and linear estimation algorithm, reports an avail-bw value which is approximately equal to the actual avail-bw minus the tool overheads (reported vertical bars at values $A-240N$kbps).
- Pathload shows very bad performance with few overlapping measurements. Interestingly, most of estimates are concentrated around certain specified values (10, 5, 7.5, 2.5 Mbps, etc). The reason behind this lays, in the binary-search process used by Pathload: depending on the interference, the tool took wrong decisions at different steps of the decision process, converging to the similar values.
- pathChirp seems to deviate less from the actual avail-bw even when several measurements overlap. The figure shows that some of the measurements are severely affected (with a worst case error above 80%) but the majority showed less bias. Since the tool automatically computes the upper and lower rates U and L, some pathChirp instances erroneously reduce the upper rate, causing the underestimation. This holds true especially in the experiments with 10 and 20 hosts.

We repeated these experiments with different avail-bw values. This way we can study the sensitiveness of the tools at different levels of link load, verifying if the accuracy of the tools changes with different avail-bw values. In Fig. 4, we compare the three methods together. Due to space constraints, we show here the results obtained with 20 active hosts and refer the reader to [4] for additional results. The results are quite surprising: while Spruce always under estimates the avail-bw by a fixed 3-4 Mbps (which approximately corresponds to the tool overhead), Pathload and pathChirp become insensitive to the actual avail-bw changes.

We have also tested all the three tools simultaneously, running 9 hosts for each different technique, with a total of 27 senders interfering. The result in a scenario without cross-traffic ($A = 10$ Mbps) is shown in Fig. 5. The measurement tools severely underestimate the avail-bw. Note that 9 Spruce measurements plus 9 pathChirp measurements alone consume almost 5 Mbps. With the addition of the Pathload hosts, it is clear that the measurement overhead becomes unsustainable, with probes consuming almost all the capacity of the link. On the other hand, if we consider the measurement traffic as "in-band", e.g. if the probes are piggy-backed on data traffic, then the tools do not perform much better: on average, pathChirp reports an avail-bw which is higher than 6 Mbps and Spruce shows more than 4 Mbps. However, since different tools have different measurement durations and since Pathload varies the transmission rate significantly, deeper investigations are required to understanding which tool performs best. We are currently undertaking several measurement campaigns to asses this problem.

[3]In all the experiments, Pathload provided upper and lower bounds very close to each other. For easiness of presentation, we consider the average between the two in our results and refer the reader to [4] for more details.

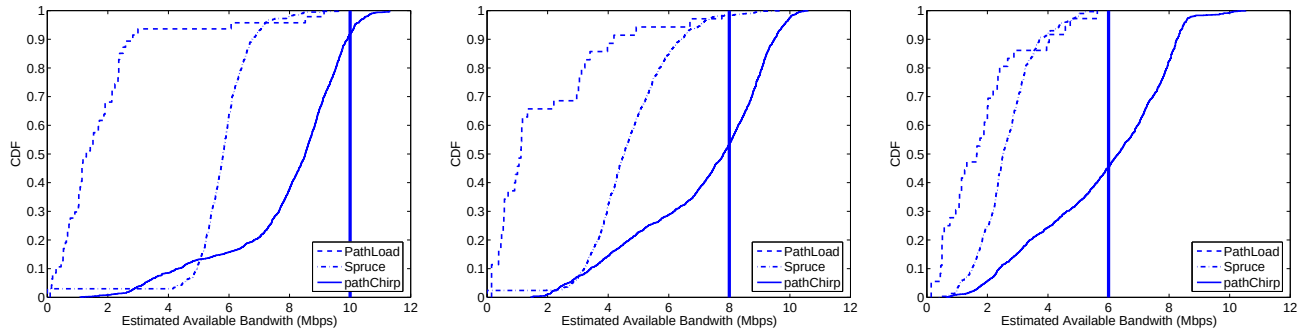


Figure 4: Sensitivity to traffic. Pathload and pathChirp become almost insensitive to bandwidth changes.

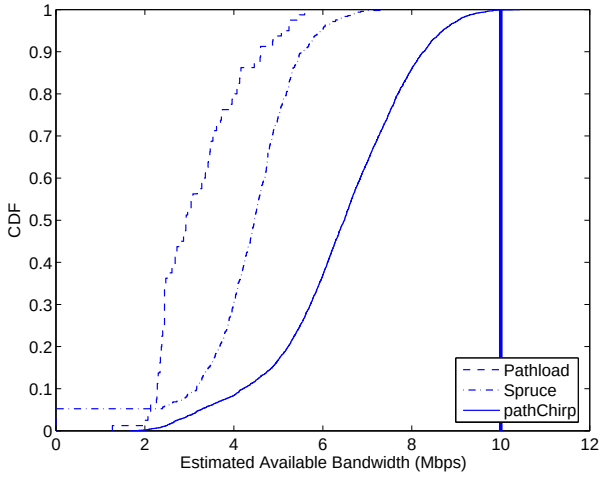


Figure 5: All tools interfering together.

4. DISCUSSION

From the presented results, it is clear that current techniques cannot be applied in large-scale distributed systems as they are. All tools have shown to underestimate the avail-bw significantly and to introduce an unacceptable overhead. To reduce the overhead, an alternative to injecting measurement packets is to piggy-back the probes on ongoing data flows, i.e., to shape "normal" traffic in packet pairs, trains or chirps depending on the chosen technique. This eliminates the overhead problem provided that there is enough data to be sent on the desired path. However, the problem of the interference between coexisting measurements is unchanged. In particular, from the results shown in the previous section, the sophisticated technique used in pathChirp seems not robust enough to tolerate multiple overlapping chirps and the exponential change in rate is probably too brutal to be employed at large. Pathload can not be used as well because the binary-search algorithm leads to unpredictable errors in presence of interference. Also, shaping the traffic in relatively long trains of various rates might add too much delay, especially for real time applications such as video streaming. Spruce seems to maintain a certain coherence in the results: if the other probes are considered as actual traffic, then the estimated avail-bw is fairly accurate, i.e., it reflects the "avail-bw" left free by the total probing process. This suggests that if the probing is performed as "in-band" process, it can be successfully exploited. For example, shaping actual data packets so as to inject packet-pairs should not be an issue, but injecting additional pure probing packets causes biased results and congestion.

In general, the results showed by Spruce suggest that avail-bw measurements in large-scale systems are *possible* provided that the measurement traffic is (i) piggy-backed as much as possible on existing data flows, (ii) the traffic shape is "smooth", with minimum impact on the network flows, and (iii) the inference algorithm is direct (PGM) or needs less iterations possible. Regarding the use of the PGM model, however, the possible inaccuracies pointed out in [12] remain unchanged. Using longer train instead of only two packets, as suggested in [13], could increase the accuracy of PGM tools but might have a higher impact on the network, as several packets would be sent at full capacity rate. In this sense, there is probably some research space for finding an optimal network operating point. Finally, it would be interesting to develop new hybrid techniques able to empower both the lightness of the PGM-model and the robustness of the PRM model.

5. CONCLUSION

In this paper, we raised the problem of interference between coexisting avail-bw measurements. We showed that current tools will *not* work in large-scale distributed systems. We studied the properties of three prominent measurement tools and discussed their functioning, their overhead and their performance in case of concurrent measurements. We used a dedicated testbed composed of 62 measurement hosts to analyze the bias introduced and we showed that all existing methods severely underestimate the avail-bw. Finally, we discussed some principles for estimating the avail-bw in large-scale distributed platforms.

6. REFERENCES

[1] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *ACM SIGMETRICS '03*, San Diego, USA. June 2003.

[2] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: architectural issues and performance. In *ACM SIGCOMM '00*, Stockholm, Sweden, Sept. 2000.

[3] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Fast Available Bandwidth sampling for ADSL links: rethinking the estimation for larger-scale measurements. In

Passive and Active Measurement Conference (PAM '09), Seoul, South Korea, Apr. 2009.

[4] *www.telematica.polito.it/croce/hotmetrics09TR.pdf*

[5] C. Dovrolis and M. Jain. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *ACM SIGCOMM '02*, Pittsburgh, USA, Aug. 2002.

[6] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *INFOCOM '01*, Anchorage, Apr. 2001.

[7] K. Hanna, N. Natarajan, and B. Levine. Evaluation of a novel two-step server selection metric. In *ICNP '01*, Riverside, USA, Nov. 2001.

[8] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC*, 21(6):879–894, Aug. 2003.

[9] M. Jain and C. Dovrolis. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In *Internet Measurement Conf. (IMC '04)*, Taormina, Italy, Oct. 2004.

[10] M. Jain and C. Dovrolis. Path selection using available bandwidth estimation in overlay-based video streaming. *Comput. Netw.*, 52(12):2411–2418, 2008.

[11] V. Konda and J. Kaur. RAPID: Shrinking the Congestion-control Timescale. In *INFOCOM 2009*, Rio de Janeiro, Brazil, Apr. 2009.

[12] L. Lao, C. Dovrolis, and M. Y. Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. In *ACM Comput. Commun. Rev.*, 36(5):29–34, 2006.

[13] X. Liu, K. Ravindran, and D. Loguinov. A stochastic foundation of available bandwidth estimation: Multi-hop analysis. *IEEE/ACM TON*, 16(1):130–143, Feb. 2008.

[14] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Conf. (PAM '03)*, San Diego, USA. 2003.

[15] A. Shriram and J. Kaur. Empirical evaluation of techniques for measuring available bandwidth. *INFOCOM 2007*, Anchorage, USA, May 2007.

[16] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and K. C. Claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *PAM '05*, Boston, USA. 2005.

[17] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Internet Measurement Conf. (IMC '03)*, Miami, USA, Oct. 2003.

[18] G. Urvoy-Keller, T. En-Najjary, and A. Sorniotti. Operational comparison of available bandwidth estimation tools. In *ACM Comput. Commun. Rev.*, 38(1):39–42, 2008.

[19] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider portal for applications. In *ACM SIGCOMM '08*, Seattle, Aug. 2008.