BET: A Hybrid Bandwidth Estimation Tool*

Alessio Botta, Salvatore D'Antonio, Antonio Pescapé, and Giorgio Ventre†

Abstract

In this paper we propose a tool that aims to integrate and improve existing tools for the available bandwidth estimation. We discuss our proposal analyzing each component of our architecture. Results from experimental analysis of our architecture and several comparative analysis with other existing and spread used tools are shown. Finally, overall conclusions are discussed and indications for future improvements are given.

1. Introduction

In the field of network monitoring, inferring the unused capacity or *Available Bandwidth (AB)* is of great importance for various network applications, for both Network Operators and End-Users (i.e. plan network upgrades, check the efficiency of applications, network aware applications, peer-to-peer file distribution and applications, server selection, SLA and QoS verification, traffic profiling, End to End admission control, congestion control and TCP, routing, Overlay Networks, multicast routing, Traffic Engineering, and finally intrusion detection and in general anomaly detection). Obtaining useful estimates of the available bandwidth from routers is often not possible due to various technical and privacy issues or due to an insufficient level of measurement resolution or accuracy. Thus, it becomes necessary to infer the required information from the network edge via an active or passive probing scheme, without requiring access to network elements or administrative resources. Our approach aims to estimate available bandwidth along a network path without access to any component along the path and without stressing existing traffic with a large volume of testing traffic. In a first stage, we can summarize the following metrics for a bandwidth (BW) estimator: (i) Total Probing Traffic; (ii) Attainable accuracy; (iii) Total estimation time. During our studies a number of tests over a broad range of bandwidth estimation tools, scenarios and experimental conditions was performed. These tests have been conducted with the aim to develop a set of practices, procedures and tools for the comparative analysis of active bandwidth estimation techniques and tools. After this study we found a lack of complete and robust bandwidth estimators according to the previous metrics. Stepping from studied tools we proposed an hybrid platform based on the effective combination of PTD (*Packet Train Dispersion*), SLoEC (*Self Loading of Exponential Chirp*), and SLoPS (*Self Loading of Periodic Streams*) methodologies able to compute the capacity and available bandwidth of network paths. In our measurement tool we use one-way techniques and we effectively combine several methodologies sending packets and waiting for replies from target nodes using UDP (in the case of data channel) and TCP (for the control channel) protocols. We called our hybrid platform *BET* (*Bandwidth Estimation Tool*). It is a receiver based application and presents an architecture composed of several functional blocks working in cascade mode. The communication between sender and receiver is carried out thanks to a novel protocol named *BEP* (*Bandwidth Estimation Protocol*).

The rest of the paper is organized as follow. In Section 2 we present a brief overview of related work. Section 3 presents BET architectural details. In Section 4 our experience in the performance analysis is described whereas in Section 5 a summary of results is presented. Finally, in Section 6 we present the ongoing work.

2. Related Work

As for related work we present an overview of mainly used applications for available bandwidth estimation.

Pathload [1] implements the SLoPS methodology. It requires access to both ends of the path, but does not require superuser privileges because it only sends UDP packets. Pathload reports a range rather than a single estimate. The center of this range is the average available bandwidth during the measurements, while the range itself estimates the

* This work has been partially supported by the Italian Ministry for Education, University and Research (MIUR) in the framework of the WEB-MINDS FIRB Project, by Regione Campania in the framework of "Centro di Competenza Regionale ICT", and finally by the E-NEXT IST European project.

† A. Botta and S. D'Antonio are with the Consorzio Interuniversitario Nazionale per l'Informatica, Naples (Italy), {abotta,sdantonio}@napoli.consorzio-cini.it. A. Pescapé and G. Ventre are with the Dipartimento di Informatica e Sistemistica, University of Napoli "Federico II", Naples (Italy), {pescape,giorgio}@unina.it.

variation of available bandwidth during the measurements. *Pathchirp* [2] uses the self inducted congestion paradigm. It sends exponential flight pattern of probes (called chirps) for causing the self induced congestion on the network. By rapidly increasing the probing rate within each chirp, *Pathchirp* obtains a rich set of information from which to dynamically estimate the available bandwidth. In [3] *netest* is introduced. It provides information to achieve better throughput while fairly sharing the available bandwidth, thus reducing misuse of the network. IGI [4] uses available bandwidth estimation techniques similar to SLoPS, but using different packet stream patterns and focusing on reducing measurement latency. A tool that is able to locate the tight link (links with available bandwidth less than of all the links preceding them) on end-to-end network paths is *Spatio-Temporal Available Bandwidth estimator (STAB)* [5]. It uses special chirp-probing trains, featuring an exponential flight pattern of packets, which have the advantage of employing few packets while giving an accurate estimate of available bandwidth.

3. BET Architecture and Modules

After analyzing existing tools we found a lack of complete and robust bandwidth estimators. We propose a tool, named BET, that compared with previously cited tools represents an effective combination of several techniques aiming to exploit the positive aspects of each of them. Also, it presents more accurate timestamping techniques.

In Figure 1 the main modules constituting the BET architecture are summarized. As shown in Figure 1 BET is constituted by four modules. The first one controls and coordinates the operations of the remaining that are responsible of the measurement process.

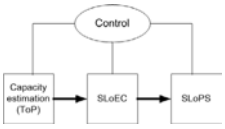


Figure 1. BET Modules

As previously said, BET is a platform based on a hybrid methodologies. More precisely, BET integrates the following the different measurement techniques: (i) the '*packet train dispersion*' [6] used for the capacity estimation; (ii) an efficient combination of the '*Self Loading of Exponential Chirp*' (SLoEC) [2] and '*Self Loading of Periodic Stream*' (SLoPS) [7] used for the available bandwidth estimation.

In particular, the dispersion of packet train provides the Asymptotic Dispersion Rate (ADR) that has a gaussian dispersion around its average value. This value represents an estimation of the real value of path capacity when there is no cross traffic. When cross traffic is present this value is less than capacity and more than available bandwidth. The ADR value found in this phase is used as an input parameter to the SLoEC module that, by using this parameter as an upper bound, makes a first estimation of the available bandwidth. To determine the duration of SLoEC phase, a set of experiments over different scenarios was done. After a large number of tests in the tuning phase, a duration of $20s$ has been chosen. In this way we achieved an average accuracy equal to 15%. This value is sufficient for our purpose because the availbale bandwidth estimated in this phase is used just as the initial value for the SLoPS phase. Indeed, starting from an initial value close to the real value the SLoPS module converges to the bandwidth estimation obtaining much more accuracy in a much shorter time period. Also, this architectural choice implies a much lower probing traffic injected in the network.

3.1. Dynamic bandwidth control

During the SLoPS phase several fleets of packet are sent from sender to receiver. Each fleet is composed of 12 flows of packets. In SLoPS implementations presented in literature, when each fleet arrives at destination the trend of received packets is calculated. This operation is carried out in order to evaluate the rate of the next fleet to send. There are two commonly used tests for the calculation of the trend: the Pairwise Comparison Test (PCT) and the Pairwise Difference Test (PDT) that are based on two different characteristics of trends. In our application a sort of dynamic bandwidth control is implemented. Hence, during each fleet, if the trend of four consecutive flows is equally estimated for both PCT and PDT tests, the fleet is interrupted. Based on the arrived flows trend, the new probing bit rate is calculated and the next fleet is sent. This dynamic bandwidth control has two main (highly coupled) advantages: (i) The probing traffic injected in the network is lower than a static choice; (ii) The time taken for measurement purpose is shorter than a static choice. To the best of our knowledge other SLoPS implementations do not support this kind of dynamic bandwidth control and they have always to send all flows in a fleet. Our approach allows for the reduction of useless traffic and guarantees an optimization in the time consumption.

3.2. The Signaling Protocol

Figure 2 shows the operations of the different modules composing BET. In Figure 2 the coordination task done by the control module is also present (drawn with discontinuous line). This module presents a client/server architecture and it controls and coordinates the operations involved in the measurement process. It uses a TCP channel. This is for

ensuring the control plane reliability. Due to space limitation we do not dig into the details of the adopted architectural choices and their implications. We just point the attention on how BET aims to improve the overall accuracy through a better evaluation of received packet timestamps (Section 3.3) and packets sent with more accurate timing (Section 3.4).

3.3. Packets arrival time estimation

3.3.1. Taking the time at user level In nearly all applications for IP metrics measurement the packets arrival time is taken at user level, by means of function *gettimeofday()*. Such a time is affected by the *gettimeofday()* function latency and for this reason it does not represent the packet arrival time at the Network Interface Card (NIC). By getting the time in such a way the accuracy in the available bandwidth estimation is jeopardized. Indeed, the function latency is not constant. It can vary among different function instances and therefore it introduces a random error in the measured arrival time.

Also, in the field of measurement applications, another issue to be addressed when using *gettimeofday()* is related to its average latency when it is used as a threshold for the detection of the context switch phenomenon. In measurement applications the context switch causes an other kind of error: the packets arrived at the NIC cannot be immediately passed to a suspended application but they have to be stored in the drivers' buffer. When the process is resumed it retrieves all arrived packets from the buffer and, thinking that they are just arrived from the sender, it gets arrival time by calling *gettimeofday()* function. In this way, starting from the second packet in the buffer, the arrival time represents just the *gettimeofday()* function latency. Furthermore, there is no indication of the real reception time. For this reason, in nearly all studied applications an algorithm for context switch detection is implemented.

3.3.2. Taking the time at kernel level To skip all the issues related to the use of the *gettimeofday()*, BET calculates the arrival times in *a different way at a different level*. We use *IOCTL()* function at kernel level. In BET, at kernel level, time stamps of arrived packets are taken by using the NIC driver. It informs the kernel about the time stamp of the last arrived packet. The *IOCTL()* function is used in order to read the driver information about the last arrived packet time (Figure 3(a)). For obtaining such an information, the *IOCTL()* is called with '`SIOCGSTAMP`' command, with such a command the time stamp of last received packet is passed from the driver to the application in the argument field. Finally, the measure of the time by using this technique is more accurate because it does not suffer of *gettimeofday()* latency and it permits to obtain a more efficient threshold value for the detection of context switch phenomenon.

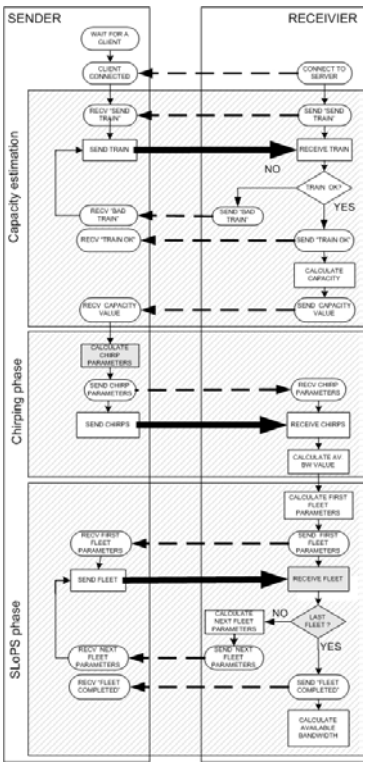
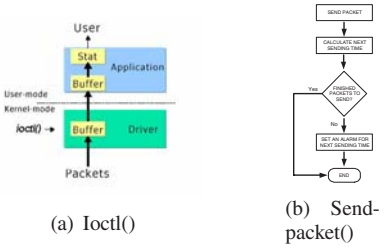


Figure 2. Data plane (continuous line) and control plane (discontinuous line)



(a) Ioctl() (b) Send-packet()

Figure 3. Time Management Functions

3.4. The packets sending process

To achieve better accuracy in the measurement process, it is important to send packets with as much as possible achievable accuracy. BET uses a technique for packets sending based on the *signal handling*. In this way it has a precision (in the time evaluation) at kernel level. Therefore, in order to send packets with the right time spacing, BET uses a packet sending function called *Sendpacket()*.

The *Sendpacket()* represents the handler of the '*UALARM*' signal. When the alarm occurs the *Sendpacket()* is called and the packet is sent. After the transmission of the packet the *Sendpacket()* sets also the alarm for the transmission of the next packet. In Figure 3(b) a block diagram of the *Sendpacket()* behavior is shown. By using the *Sendpacket()* function we have created a recursive method for packets sending with high accuracy in the time spacing. Finally, in addition to the kernel time accuracy, the utilization of *Sendpacket()* for the packet sending, improves BET with of other two main advantages: (i) In the packet sending phase BET has an active socket used for the signaling process; (ii) During packets sending by using the *Sendpacket()* function, if a '*context switch*' occurs, the function is not suspended (due to the association to a signal). In this way the packet sending process does not suffer of the delay caused by context switch operations.

4. Experimental Analysis

This analysis has been conducted by using `BET` and comparing it to other three spread used available bandwidth tools: *Pathload*, *Netest*, and *Pathchirp*. These tools were tested with different cross traffic (CT) profiles at different bit rates. Indeed, thanks to the use of a powerful traffic generator - *D-ITG* (*Distributed Internet Traffic Generator*) [8] - we are in charge of using a large number of random variables to profile inter departure time (IDT) and size (PS) of cross traffic packets. In Figure 4 the controlled test-bed used in our tests is shown. The link under test is located between the routers $Psinoe$ and $Telsiope$. It is an Ethernet link with nominal capacity equal to $100Mbps$. In order to have a first precise evaluation of the tools performance, we used as a proof-of-concept a controlled and fully configurable open test-bed. In this way we were in charge of control as much variables as possible as well as to configure to our own several network topologies. Also, in this first stage, we preferred to use a controlled test-bed to have a fully control on the network devices and network traffic (both active and cross traffic). Before to step into experimental details, it is worth mentioning that we repeated each test several times. The values reported in the following graphics and tables represent a mean value across three test repetitions. Thanks to the use of controlled test-beds we had a confidence interval greater than or equal to 95%.

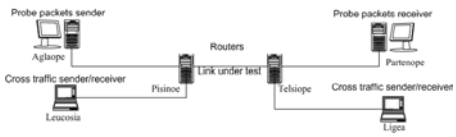
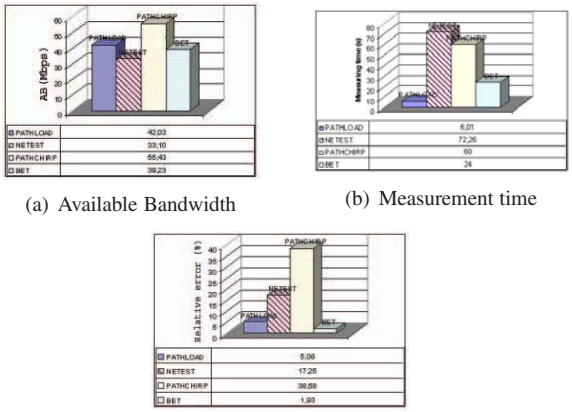


Figure 4. Our test-bed

In Figure 5 the results of one test are depicted. In such a case we generated CT with constant IDT and constant PS. The CT bit rate was equal to $60Mbps$, therefore the expected available bandwidth was equal to $40Mbps$. In Figure 5 the estimated available bandwidth (5(a)), the relative error (RE) of available bandwidth measure [1] (5(b)), and the total measurement time (5(c)) are presented. In Figure 5 we can see that the best performance in terms of accuracy were obtained by `BET`. It shows a relative error less than 2% achieved with a masurement time of $24s$.



(a) Available Bandwidth

(b) Measurement time

(c) Relative error

Figure 5. UDP constant cross traffic ($60Mbps$)

Due to space limitations we can not provide graphical results for all conducted tests. For this reason, in Table 1 other interesting results are summarized. Furthermore, other considered scenarios will be analyzed by referring to the case depicted in Figure 5 as the 'reference case'.

Digging into details of Table 1, in the case of constant profile cross traffic (*constant PS* and *constant IDT*) with bit rate equal to $30Mbit$, we obtained the same results of the reference case. Indeed, also in this test `BET` achieved the best performance in terms of accuracy. Instead, when the cross traffic rate was equal to $90Mbit$, `Pathchirp` achieved the best performance in terms of accuracy. In all the tests with constant profile cross traffic, `Netest` showed the worst behavior in terms of both total measurement time (TMT) and accuracy. For this reason `Netest` has not been used for successive tests. Furthermore, when cross traffic was generated with *Pareto PS* and *exponential IDT* the best performance in terms of both measurement time and accuracy was obtained by `Pathchirp`. In the case of *Poisson PS* and *exponential IDT*, `Pathchirp` showed the best per-

1 $RE = \frac{EV-MV}{EV}$ where RE is the relative error, EV is the expected value, and MV is the measured value.

formance in terms of accuracy while the worst in terms of measurement time. Finally, in the case of cross traffic with *exponential PS* and *Poisson IDT*, BET achieved the best performance in terms of accuracy while the worst in terms of measurement time.

CT (Mbps)	PS	IDT	Parameter	BET	Path Load	Path Chirp	NetEst
90	Constant	Constant	**AB** (Mbps)	5.72	4.87	10.42	5.86
			RE (%)	42.85	51.25	4.20	41.42
			TMT (s)	50.6	43.97	10	2.25
60	Constant	Constant	**AB** (Mbps)	39.23	42.03	55.43	33.10
			RE (%)	1.93	5.08	38.58	17.25
			TMT (s)	24	6.01	60	72.26
30	Constant	Constant	**AB** (Mbps)	69.2	71.42	90.12	73.94
			RE (%)	1.14	2.03	28.74	5.62
			TMT (s)	22.7	6.20	10	74.79
32	Pareto	Exponential	**AB** (Mbps)	52.22	38.15	57.83	
			RE (%)	23.21	43.90	14.95	
			TMT (s)	47.3	35.8	10	
50	Poisson	Exponential	**AB** (Mbps)	42.23	36.39	56.46	
			RE (%)	15.54	27.22	12.92	
			TMT (s)	23.32	7.35	30	
50	Exponential	Poisson	**AB** (Mbps)	38.78	35.88	61.91	
			RE (%)	22.44	28.25	23.82	
			TMT (s)	20.2	9.92	10	

Table 1. Experimental results and comparative analysis under different traffic profiles

5. Conclusion

Based on our practical experience there is not a tool better than others for the available bandwidth estimation. We believe that this research topic represents a fertile field where there is still room for many innovative approaches, also considering the case of wireless links. We tested several tools and we compared the performance of our proposed platform. We learned that many of the current proposals are dependent from the network status and cross traffic. Thanks to a deep experimental analysis performed by using a large number of tests over a controlled test-bed, this our seminal work do not give us the possibility to choose the best tool for all network conditions. At opposite side, we are in charge of understand the behavior of the spread used tools in the case of several traffic patterns (constant, exponential, poisson, pareto, ...) and we prove that our proposal represents a good compromise among considered variables in a fitness function that considers accuracy, total estimation time, and total probing traffic. Indeed, as for these variables we tested many network conditions and we summarize our goals presenting experimental results by using D-ITG as cross traffic generator. Thanks to the use of this platform we were in charge of clearly define the sent and received cross traffic.

6. Between Current and Future Work

The achieved results have shown that, in many cases, our application performs better than other ones. But, in general, does not exist '*the best tool for the available bandwidth measure*'. After this deep analysis we think that a great effort has been done but other contributions are needed. Currently, we are working on a modification to the SLoPS algorithm in order to improve the performance of the measurement process. Due to the adopted cascade model the measurement time is, in some cases, larger than the Pathload one. For this reason an envisaged improvement is the reduction of the chirping phase duration by adopting a dynamic approach. More precisely, according to a cross traffic estimation chirping phase duration could be chosen in an automated fashion. By taking into account the results of our experimental analysis and after a deep analysis of the SLoPS algorithm, we are planning to lightly change the available bandwidth scanning technique used in the SLoPS module of BET. In our next implementation we will use a different technique that we called *SAWT*. SAWT consists of sending (within the same fleet) flows with increasing bit rate. In this way a more efficient scan of available bandwidth is performed as shown in Figure 6.

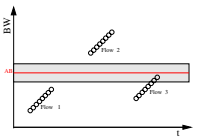


Figure 6. SAWT technique

References

[1] M. Jain, C. Dovrolis, "Pathload: a Measurement Tool for Available Bandwidth Estimation", *Proc. of* Passive and Active Measurement Workshop, 2002.

[2] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, L. Cottrell. "PathChirp: Efficient Available Bandwidth Estimation for Network Paths", *Proc. of* PAM, 2003.

[3] G. Jin, B. Tierney, "Netest: A Tool to Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput", *Proc. of* The 8th IFIP/IEEE International Symposium on Integrated Network Management March 24-28, 2003.

[4] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques", IEEE JSAC, Journal on Selected Areas in Communications, Vol. 21, No. 6, August 2003.

[5] V. J. Ribeiro, R. H. Riedi, and R. G. Baraniuk, "Locating Available Bandwidth Bottlenecks", IEEE Internet Computing, September 2004, pp. 34-41.

[6] R. L. Carter and M. E. Crovella "Measuring bottleneck link speed in packet-switched networks", Performance Evaluation Journal, October 1996.

[7] M. Jain, C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics and Relation with TCP Throughput" *Proc. of* ACM SIGCOMM, 2002.

[8] http://www.grid.unina.it/software/ITG