Performance Analysis of Available Bandwidth Estimation Tools for Grid Networks

Daniel M. Batista*, Luciano J. Chaves*, Nelson L. S. da Fonseca*, and Artur Ziviani†
*Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil.
Emails: batista@ic.unicamp.br, luciano@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br
†National Laboratory for Scientific Computing (LNCC), Petrópolis, Brazil.
Email: ziviani@lncc.br

***Abstract*—Modern large-scale grid computing for processing advanced science and engineering applications relies on geographically distributed clusters. In such highly distributed environments, estimating the available bandwidth between clusters is a key issue for task scheduling. We analyze the performance of two well known available bandwidth estimation tools, `pathload` and `abget`, with the aim of using them in grid environments. Our experiments consider the accuracy of the estimation, the convergence time, their level of intrusion in the grid links, and the ability to handle multiple simultaneous estimations. Overall, `pathload` represents a good solution to estimate available bandwidth in grid environments.**

I. Introduction

Modern large-scale grid computing for processing advanced science and engineering applications relies on geographically distributed clusters [1], demanding coordinated resource sharing for problem solving in heterogeneous dynamic environments. Grid computing differs from conventional parallel computing since the latter involves confined systems and uses local networks for data transfer, whereas the former is composed of geographically distributed clusters interconnected via a wide area network with communication links belonging to different administrative domains and shared by large number of applications.

In parallel/distributed computing, applications are divided in logical executable pieces called tasks. Grid scheduling involves the assignment of tasks to hosts and their starting execution time is influenced by host characteristics such as CPU and memory capacity as well as by network characteristics such as delay, packet loss rate, and bandwidth.

Since grid computing relies on the availability of bandwidth for data transfer among distributed tasks, estimating it is a key issue for task scheduling. The available bandwidth in network links is highly dynamic and thus needs to be frequently measured for the production of efficient grid schedules. Given a certain time interval, measurements of the available bandwidth have inherent uncertainties due to its dynamic nature and the inaccuracy of the adopted estimation tools [2]. These uncertainties justify that available bandwidth estimations be represented by intervals rather than as simple averages. Considering the available bandwidth as a single value and ignoring existing uncertainties can increase the makespan of grid applications by roughly 20%, as shown in [3]. Such uncertainties influence the tuning of data transfer control. Results indicate that the automatic adjustments of `GridFTP` parameters are directly related to the available bandwidth and the bandwidth-delay product between processing nodes [4].

The contribution of this paper is to investigate the adequacy of existing available bandwidth estimation tools for their adoption in the scheduling of grid tasks. Criteria for comparing these tools are: accuracy of estimation, convergence time, level of intrusion in grid links, and ability to handle simultaneously multiple estimations. Convergence time impacts the makespan of grid applications and it should be relatively short compared to the expected makespan of the application since the time to produce a task scheduling should also be short [5]. The level of intrusion in grid links impacts resource availability to other users since communication links are shared resources. Moreover, analyzing the ability to perform simultaneous estimations is quite relevant since this is common in operational grids given the asynchronous nature of users and applications.

We consider the `pathload` [2] and `abget` [6] tools for available bandwidth estimations since among the various estimation tools (e.g. see [6], [7]), only these tools provide intervals associated to their estimations. Other studies that compare tools for estimating available bandwidth including `pathload` and `abget` can be seen in [8]–[10]. Nevertheless, previous works focus on the accuracy of estimations and disregard, at least, one of the other criteria adopted in this paper which are of paramount importance to large-scale grid environments. Results derived point out the `pathload` as the preferred tool for adoption in grid environments.

The remainder of this paper is organized as follows. Section II briefly overviews the process of bandwidth estimation. In Section III, the available bandwidth estimation tools `pathload` and `abget` are presented. Section IV describes the experiments performed as well as discusses the results obtained. Conclusions are drawn in Section V.

II. Bandwidth Estimation

Estimating bandwidth includes measuring different metrics [11] such as link nominal capacity, bottleneck capacity along a path, end-to-end available bandwidth along a path, and bulk transfer capacity (BTC) between pairs of hosts. Fig. 1 illustrates these metrics. Host 1 and Host 2 are interconnected by a path composed of three links, depicted as rectangles, with the gray part representing the used capacity and the white one

the available bandwidth. The nominal capacity of each link is $C1$, $C2$, and $C3$ and the bottleneck capacity of the path is $C1$, thus, the end-to-end available bandwidth is $A3$. BTC is the maximum throughput obtained by a TCP connection along the network path. We cannot represent the BTC metric in Fig. 1 because it depends on the type of concurrent traffic found in the path links at the moment measures are taken.

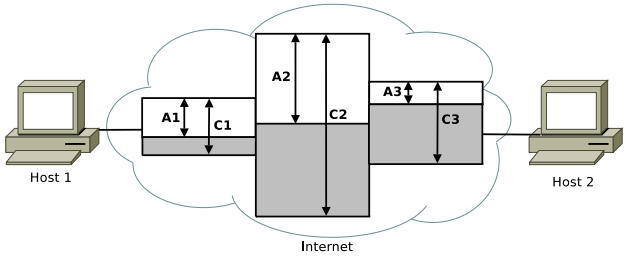

Figure 1. Illustration of bandwidth metrics (based in a figure from [11]).

In this paper, the focus is on tools to estimate end-to-end available bandwidth. Evaluation of tools for estimating BTC is not carried out since not all grid applications use the TCP protocol to transfer data, such as multimedia applications in CineGrid [12], a grid that produces and transmits high definition digital media using processing and communication resources spread worldwide.

III. Tools for Available Bandwidth Estimation in Grid Networks

The `pathload` and `abget` tools perform their estimations using the Self-Loading Periodic Streams (SLoPS) technique, proposed in [2]. In this technique, several sequences of packets are transmitted between two end nodes using different rates, for measuring the One-Way Delay (OWD) [1] of every packet sent at different adopted rates. If an increase of the OWD is detected, it means that the corresponding transmission rate is larger than the available bandwidth; otherwise, the transmission rate is smaller than the available bandwidth. Information on variations of OWD value is exchanged between end nodes to estimate an interval width that represents the end-to-end available bandwidth along the path between the two end hosts.

The tools analyzed differ by the way they implement the SLoPS technique. `pathload` requires processes to be executed in both end hosts in order to allow the exchange of information about the OWD value measured from source to destination. `abget` executes at one end host but it requires a web server (HTTP) either at the other end host or in the local network this host is located. The control of the transmission rate of the packets and the monitoring of the OWD in `abget` are performed by the source itself by using data related to the control algorithms of TCP.

[1]To measure OWD, both end nodes have to be somehow synchronized. Possible solutions for this issue might be, at different level of synchronization accuracy, the adoption of NTP (Network Time Protocol) servers, the use of GPS cards at both ends, or a software clock to enhance measurement accuracy without using GPS cards [13].

The `pathload` tool works as follows. To estimate the available bandwidth from host A to host B using `pathload`, there's a need to execute a "sender" process at host A and a "receiver" process at host B. Information concerning the variations of OWD value is exchanged between end hosts via a TCP control connection. The "sender" initiates the estimation process sending one sequence of UDP packets at an initial rate R_{start}. As the packets arrive at the "receiver", the OWD value is computed. In case no increase of OWD value is observed, the "receiver" notifies the "sender" to increase the packet sending rate. At the end of the estimation, `pathload` provides as output an interval $[R^{min}, R^{max}]$ that corresponds to the range of available bandwidth along the path between hosts A and B.

To estimate the available bandwidth from host A to host B, one may launch an `abget` client at host A and direct it to a TCP server (e.g. a web server) running at host B or at a host located at the same network where host B resides. `abget` simulates the operation of TCP so that it controls the rate host A delivers packets to the host B. The `abget` client ignores the standard operating system implementation of TCP and manipulates ACKs sent to host B. Fake ACKs are sent to host B informing the acknowledgment of only 1 MSS. Upon receiving each ACK, host B sends a packet with size 1 MSS. If host A sends ACKs with a time period T, the `abget` client induces host B to send data with a rate $R = MSS/T$. Upon the arrival of packets at host A, the OWD value is measured and the rate at which ACKs are generated is adjusted in order to find the interval $[R^{min}, R^{max}]$.

If on one hand `abget` has the main advantage of providing estimations running only at one end, on the other hand, it requires administrative privilege to operate TCP differently than the standard implementation in the local operating system. Another drawback is that many parameters need to be informed manually to allow a relatively fast convergence to the interval corresponding to the available bandwidth estimation. The `pathload` tool disregards such parameters due to the TCP control connection which allows fine tuning of SLoPS in run time. However, besides involving both end hosts, `pathload` uses UDP to estimate the available bandwidth, which can be ineffective since UDP packets are commonly blocked in firewalls due to security reasons. In contrast, `abget` does not face this problem because most domains already allow the delivery of packets to HTTP servers.

IV. Performance Analysis

Two different scenarios were employed to evaluate the `pathload` and `abget` tools. The first scenario involves links of small nominal capacities (10Mbps) and the second scenario links with large nominal capacities (1Gbps). It is important to analyze these tools in both scenarios due to the heterogeneity of link capacity existing in clusters interconnection in typical grids.

The first scenario, illustrated in Fig. 2, was emulated in `NCTUns` [14] which allows the integration of simulated network topologies with real hosts running actual applications

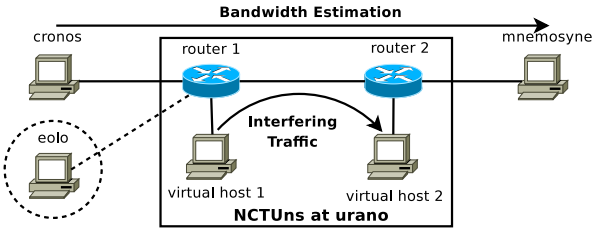

Figure 2. Experimental setup with NCTUns.

without requiring any modification of these applications. The estimations of available bandwidth in this first scenario are performed from the hosts cronos and eolo to the host mnemosyne. All these hosts are real hosts located in the same local Gigabit Ethernet network. `NCTUns` was executed in the host named urano, located in the same local network. In order to observe the behavior of the estimation tools under different network conditions, two virtual hosts have been used in `NCTUns` to generate interfering traffic aiming at interfering in the available bandwidth between the real hosts. Table I summarizes the configuration of the hosts involved in the experiments. The host mnemosyne runs the Apache web server version 2.2.3 and this allowed the execution of the `abget`. It's important to observe that the web server did not have any network resources reservation during the experiments.

Table I
CHARACTERISTICS OF THE HOSTS USED IN THE EXPERIMENTS.

Host	**Processor/Memory**	**Operating System** (Linux)
eolo	Intel Core 2 Quad 2.66GHz / 4GB	Debian kernel 2.6.23.1
cronos	Intel Core 2 Quad 2.40GHz / 4GB	Debian kernel 2.6.23.1
mnemosyne	Dual Xeon 2GHz / 4GB	Debian kernel 2.6.23.1
urano	Intel Core 2 Duo 2.13GHz / 4GB	Fedora kernel 2.6.25.9

Three metrics have been evaluated in the experiments for each tool: estimation accuracy, execution time, and level of intrusion. The first scenario involves CBR UDP traffic at 2, 4, 6, 8, and 10Mbps rates, TCP traffic and no interfering traffic. The nominal capacity of the links has been fixed in 10Mbps during measurements. In a first step, measures were collected only between the hosts cronos and mnemosyne. In a second step, measures were collected simultaneously between hosts cronos and mnemosyne and between hosts eolo and mnemosyne. Due to space limitation, only the most relevant results are presented and discussed.

Fig. 3 to Fig. 6 present the results obtained for the first step of the first scenario. Figs. 3 and 4 present results provided by `pathload` and `abget`, respectively, as a function of interfering traffic. In these figures, the curve named "Available Bandwidth" represents the actual available bandwidth between the hosts. The gray areas named "pathload estimation" and "abget estimation" show the intervals that have been provided by these tools. For each configuration of interfering traffic, the estimation tools were executed twice. Results show that `pathload` estimations are closer to the actual values when there is interfering traffic. Besides that, `abget` estimations do not follow the availability along the path between cronos and mnemosyne. With interfering traffic $\leq$ 4Mbps, `abget` estimated that the path was almost 100% available, whereas with interfering traffic $\geq$ 6Mbps its estimations indicated that the link was almost 100% unavailable. The estimation intervals from `abget` were on average larger with TCP interfering traffic than the other configurations of interfering traffic.

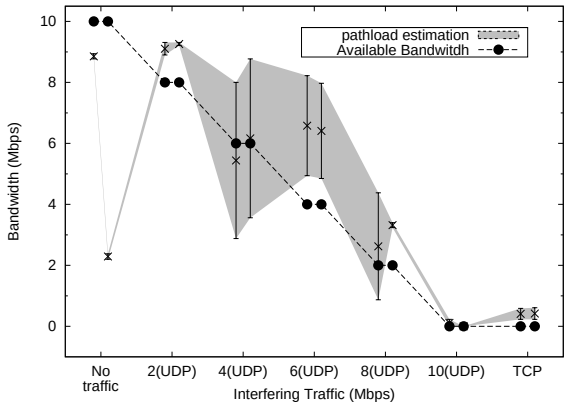

Figure 3. `pathload` estimations.

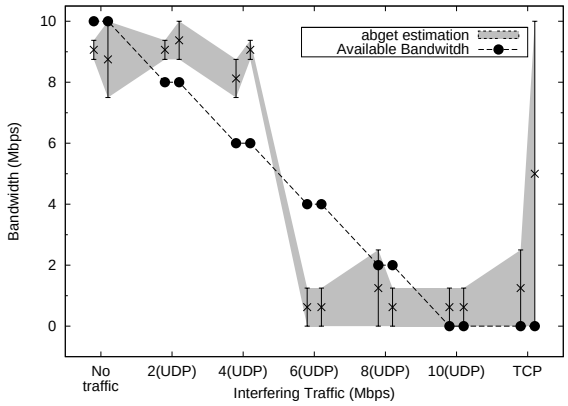

Figure 4. `abget` estimations.

The execution time of the estimation tools is shown in Fig. 5. Since `abget` does not demand an initial adjustment of the transmission rate as `pathload` does (since there is no need to provide a value close to the nominal capacity of links), it requires the transfer of the same quantity of data at the beginning of the estimation regardless of the availability of the links. As a consequence, as the links become more utilized, the execution time tends to increase, as observed in the "abget" curve. Nevertheless, when the interfering traffic is small ($\leq$ 8Mbps), `abget`, executed with the option to estimate intervals via a binary search, converges to results faster than `pathload` does. The low execution time of `abget` in one execution with TCP interfering traffic is due to the fact that the tool was not able to connect with the web server at mnemosyne because of the heavy load.

Levels of intrusion are shown in Fig. 6, expressed as the volume of injected bytes by each tool during its execution. We observe that `abget` has a more stable behavior than `pathload`. The `pathload` tool tends to reduce its generated traffic as the links get less available. This happens because the transmission rate of `pathload`, at each iteration of the

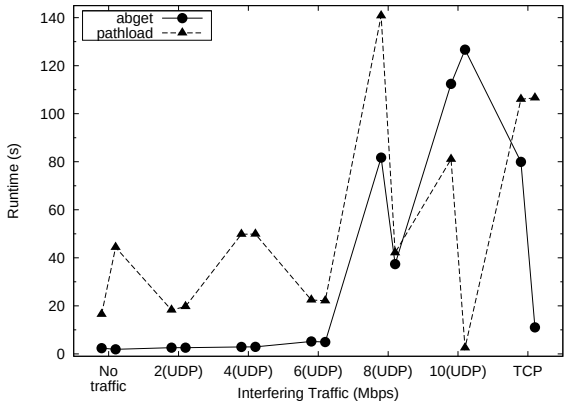

Figure 5. Execution time of the estimation tools.

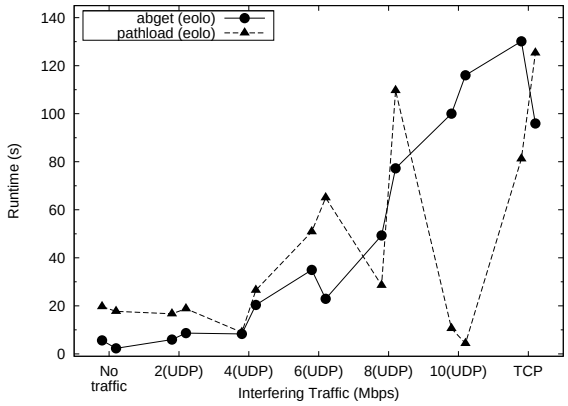

Figure 7. Execution time of tools (simultaneous execution – eolo).

algorithm, is not guided by the binary search implemented in `abget`. In this way, after some iterations, `pathload` can reduce its transmission rate and inject less traffic than `abget`.

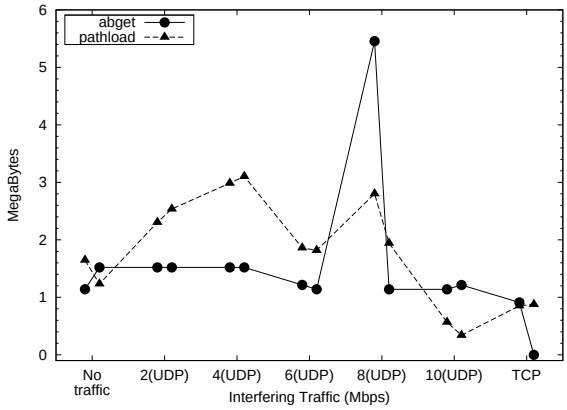

Figure 6. Level of intrusion of the estimation tools.

Some of the results obtained in the second step of the first scenario are shown in Fig. 7. Other results were similar as those obtained with just one instance of the estimation tools in execution. In order to allow the simultaneous execution of `pathload`, its source-code was modified because the ports used by the program are statically defined in the original code. The trends of the available bandwidth estimation and of the level of intrusion with two simultaneous executions were similar to that of a single execution. The execution time provides the main difference between these two types of execution. Fig. 7 shows the execution time for samples collected between the hosts eolo and mnemosyne (similar results were observed between cronos and mnemosyne). Comparing with the execution time when a single instance was executed (Figure 5), an increase in the execution time of `abget` can be observed. The combination of traffic from two instances increases the execution time of the estimation tool starting at 4Mbps (UDP) while such increase happens only at 8Mbps (UDP) when there is only one instance.

In general, in the first scenario, `pathload` provided better estimations than `abget`, although `abget` executed faster and was less intrusive. Both tools executed relatively fast ($<$ 2m20s) and with a relatively low level of intrusion ($<$ 6MB) if compared with the transfers of time and volume magnitudes expected for grid applications.

In the second scenario, the tools were evaluated without using `NCTUns` in a local network with links of 1Gbps interconnecting the hosts. The `iperf` [15] tool was used to generate interfering traffic. Virtual hosts and routers were created in the urano machine using virtual network interfaces. Estimations were performed using the same hosts of the first scenario. Although links have nominal capacity of 1Gbps, the observed actual capacity was 525Mbps, so this value was considered instead. The same metrics and steps of the first scenario were considered in the second scenario. The difference is in the rates of UDP interfering traffic (105, 210, 315, 420, and 525Mbps).

The accuracy of the tools when estimations were performed only between cronos and mnemosyne is shown in Figs. 8 and 9. Estimations derived by `pathload` (Fig. 8) were more accurate than those from `abget` (Fig. 9). In contrast to the first scenario, in the second scenario `pathload` estimations clearly follow the bandwidth availability in the links. Similarly to the first scenario, `pathload` provided more accurate estimations when there was more interfering traffic. Besides that, `abget` presented a different behavior when compared to the first scenario. Although grid links had more nominal capacity, the `abget` estimated that links were almost 100% occupied regardless of the intensity of interfering traffic. Parameter tuning was performed for `abget`, however, no better results were achieved. Actually, the need to set a high number of parameters in `abget` is a negative point, specially when under high dynamic environments such as grids.

The execution times of `abget` were on average much higher than those of `pathload` (Fig. 10). `pathload` took longer periods to execute than `abget` only under the influence of TCP interfering traffic.

The level of intrusion increased proportionally to the increase of the link capacities when compared to the results in the first scenario (Figure 6). The behavior of the tools was the same, which also happened in the case of two instances.

It is important to point out the behavior of `pathload` in the second step of the second scenario when the interfering traffic was 105Mbps (UDP). In one of the executions, this tool gen-

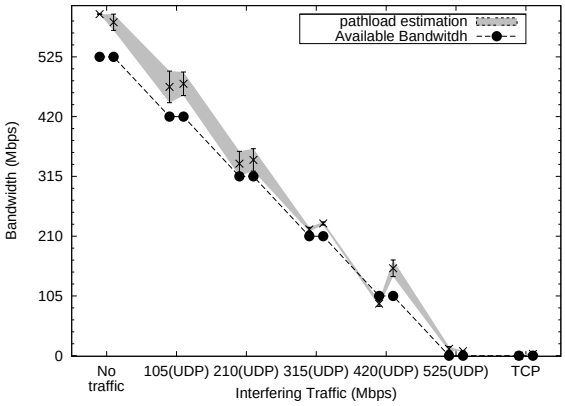

Figure 8. `pathload` estimations.

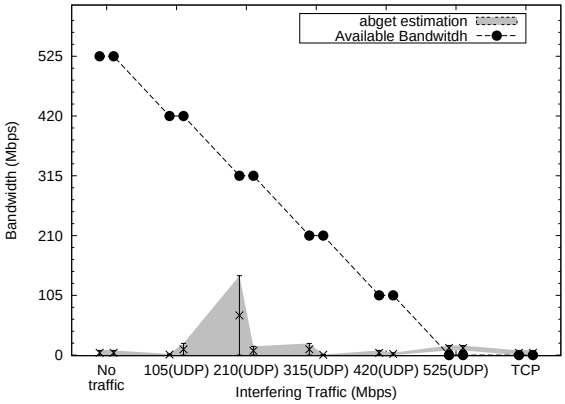

Figure 9. `abget` estimations.

erated about 300MB of data, a volume of data non-negligible. Such behavior evinces that the available bandwidth estimation tools can be highly intrusive in the network, which motivates the implementation of preventive procedures to avoid it. A simple solution would be either to request users a maximum value for the execution time or to quantify the maximum allowed value for the injected bytes. Although external stop conditions may result in less precise estimations, the overall outcome can be preferable if all the involved metrics are jointly considered.

In summary, for the second scenario, `pathload` provides good results even in an environment with high capacity links. The execution time of `pathload` was lower than that of `abget`, although `abget` was on average less intrusive. Schedulers using `abget` to estimate available bandwidth in this type of scenario would provide scheduling that leads to underutilization of network links due to the underestimation of the available bandwidth. The `pathload` tool has the potential to flood the network with traffic that may directly impact other applications in execution.

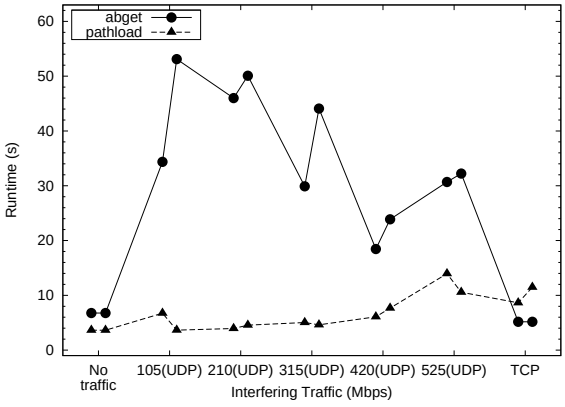

Figure 10. Execution time of the estimation tools.

V. Conclusion

Overall, considering all scenarios, `pathload` is the preferable tool for estimating available bandwidth in grid networks. Nevertheless, users and administrators of grid environments should consider the potential impact the estimations may have in other flows during measurements. Furthermore, `pathload` has to be executed in both end hosts, a negative characteristic that might motivate modification in `abget` to improve its accuracy. The code of `pathload` must also be changed so that the adopted ports are dynamically defined, allowing its simultaneous use between several pairs of end host in a large-scale distributed grid environment.

References

[1] I. Foster, "What is the Grid? A Three Point Checklist," *GRIDToday*, vol. 1, no. 6, Jul 2002, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf. Retrieved January 6, 2009.

[2] M. Jain and C. Dovrolis, "End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM TON*, vol. 11, no. 4, pp. 537–549, Aug 2003.

[3] D. M. Batista, A. C. Drummond, and N. L. S. Fonseca, "Robust Scheduler for Grid Networks," in *Proceedings of the ACM SAC'09*. New York, NY, USA: ACM Press, Mar 2009, pp. 354–358.

[4] T. Ito, H. Ohsaki, and M. Imase, "On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Grid Computing," in *Proceedings of the BroadNets 2005*, vol. 2, Oct 2005, pp. 1338–1344.

[5] D. M. Batista, N. L. S. da Fonseca, F. K. Miyazawa, and F. Granelli, "Self-Adjustment of Resource Allocation for Grid Applications," *Computer Networks*, vol. 52, no. 9, pp. 1762–1781, Jun 2008.

[6] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. Markatos, and C. Dovrolis, "Available Bandwidth Measurement as Simple as Running wget," in *Proceedings of the PAM Workshop*, 2006.

[7] Cooperative Association for Internet Data Analysis (CAIDA), "CAIDA : tools : taxonomy," Jun 2008, http://www.caida.org/tools/taxonomy/performance.xml#bw. Retrieved January 6, 2009.

[8] A. Botta, A. Pescape, and G. Ventre, "On the Performance of Bandwidth Estimation Tools," in *Proceedings of Systems Communications*, Aug 2005, pp. 287–292.

[9] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, and A. Broido, "Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links," in *Proceedings of the PAM Workshop*. Springer, Mar 2005.

[10] J. Sommersa, P. Barforda, and W. Willinger, "Laboratory-based Calibration of Available Bandwidth Estimation Tools," *Microprocessors and Microsystems*, vol. 31, no. 4, pp. 222–235, Jun 2007.

[11] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *Network, IEEE*, vol. 17, no. 6, pp. 27–35, 2003.

[12] D. Shirai, T. Kawano, T. Fujii, K. Kaneko, N. Ohta, S. Ono, S. Arai, and T. Ogoshi, "Real Time Switching and Streaming Transmission of Uncompressed 4K Motion Pictures," *Future Generation Computer Systems*, vol. 25, no. 2, pp. 192–197, Feb 2009.

[13] A. Pásztor and D. Veitch, "PC-based precision timing without GPS," in *ACM SIGMETRICS*, Los Angeles, CA, USA, Jun. 2002.

[14] S. Y. Wang, C. L. Choua, and C. C. Lin, "The Design and Implementation of the NCTUns Network Simulation Engine," *Simulation Modelling Practice and Theory*, vol. 15, no. 1, pp. 57–81, Jan 2007.

[15] NLANR, "Iperf," 2008, http://sourceforge.net/projects/iperf/?abmode=1. Retrieved January 7, 2009.