J Supercomput (2010) 53: 103–121
DOI 10.1007/s11227-009-0344-z

Performance analysis of available bandwidth estimation tools for grid networks

Daniel M. Batista · Luciano J. Chaves · Nelson L.S. da Fonseca · Artur Ziviani

Published online: 22 October 2009

Abstract Modern large-scale grid computing systems for processing advanced science and engineering applications rely on geographically distributed clusters. In such highly distributed environments, estimating the available bandwidth between clusters is a key issue for efficient 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. Differently than previous investigations (Jain et al., http://www.caida.org/workshops/isma/0312/slides/rprasad-best.pdf; Shriram et al., in Passive and active network measurement: 6th international workshop, PAM 2005. Springer, Berlin, 2005), our experiments consider a series of relevant metrics such as accuracy of the estimation, convergence time, degree of intrusion in the grid links, and ability to handle multiple simultaneous estimations. No previous work has analyzed the use of available bandwidth tools for the derivation of efficient grid scheduling.

Keywords Grids · Networks · Bandwidth estimation · Performance evaluation

D.M. Batista · L.J. Chaves · N.L.S. da Fonseca (✉)
Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil
e-mail: nfonseca@ic.unicamp.br

D.M. Batista
e-mail: batista@ic.unicamp.br

L.J. Chaves
e-mail: luciano@lrc.ic.unicamp.br

A. Ziviani
National Laboratory for Scientific Computing (LNCC), Petrópolis, Brazil
e-mail: ziviani@lncc.br

1 Introduction

Modern large-scale grid computing systems for processing advanced science and engineering applications rely on geographically distributed clusters [3], 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 a 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 the start of their execution is influenced by host characteristics such as CPU and memory capacity as well as by network characteristics such as bandwidth.

1.1 Requirements for bandwidth estimation in grids

The deployment of high capacity links enabled the emergence of grid systems since they rely on these links for the transfer of huge amount of data. Several reports ([4, 5] and [6]) emphasized the relevance of bandwidth availability for grid performance. In [4], statistics on data transfer in several grids were presented. It was reported that 55% of grid applications transfer between 1 to 100 MB and 18% transfer at least 10 GB. It is expected that in the next decade grid applications will demand transmission rates of the order of Terabits per second. The relevance of network link capacity for grid scheduling becomes evident when analyzing the reduction of the execution time of applications when the required bandwidth is available [6].

Moreover, Silvester [7] showed that the execution of highly demanding grid applications can induce a growth of five to eight times in the utilization of network links. This increase can be highly significant for emerging applications, such as the recent CERN's LHC Computing Grid (LCG) which expects to distribute and process around 15 PB per year [8]. Furthermore, such an increase brings significant fluctuations of the available bandwidth.

Since grid scheduling relies on the knowledge of the available bandwidth for data transfer among distributed tasks, accurately estimating this value is a key issue for efficient task scheduling. Moreover, the available bandwidth is highly dynamic and thus needs to be frequently measured to support efficient grid schedules. Given a certain time interval, measurements of the available bandwidth have inherent uncertainties due to both its dynamic nature as well as due to the inaccuracy of the adopted estimation tools [9]. These uncertainties justify the estimation of the available bandwidth to be represented by intervals rather than by simple averages. Considering the available bandwidth as a mean value and ignoring existing uncertainties can increase the makespan of grid applications by roughly 20% [10]. Such uncertainties also influence the efficient tuning of data transfer control. For example, recent results indicate that the automatic adjustments of `GridFTP` parameters are directly related to the available bandwidth and the bandwidth-delay product between processing nodes [11].

1.2 Contributions of the paper

Several grid systems adopt self-adjustment procedures for the allocation of resources in order to cope with fluctuation of resource availability ([12–15]). The adoption of these procedures are motivated by the fact that resources of grids are not usually owned by their users who do not have exclusive right of use of grid resources. In this approach, grid resources are monitored and if a different allocation of distributed resources leads to lower makespan, tasks are migrated to the new allocation scheme. Having accurate estimations of the available bandwidth is, thus, of paramount importance to evaluate the potential rescheduling of the tasks of an application.

This paper investigates the adequacy of existing available bandwidth estimation tools for their adoption in the scheduling of grid tasks. To the best of our knowledge, this is the first work to analyze the use of available bandwidth tools in the context of network-based high performance computing for the derivation of efficient grid scheduling.

A distinguishing aspect of the present study is the comprehensive comparison of tools for bandwidth estimation using various performance metrics. Most of previous works, which compared such tools, adopted the precision of their estimation as the main if not the solely metric used. Workload characterization, benchmarks, and metrics for grid computing have been defined in the investigations carried out so far by taking into consideration the grid operation and the descriptions of the applications [16, 17]. There is no standard suite of benchmarks and metrics defined for the communication aspects of grid networks and in particular for the estimation of the available bandwidth. Thus, we established criteria and metrics for the comparison of available bandwidth estimation tools in the present investigation which are: accuracy of estimation, convergence time, level of intrusion in grid links, and the ability to handle simultaneously multiple estimations.

We consider the `pathload` [9] and the `abget` [18] tools for estimating the available bandwidth since among the various estimation tools (e.g. see [18, 19]), 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 [1, 2]. Nevertheless, previous works focused on the accuracy of estimations and disregarded all the other metrics analyzed in this paper which are of paramount importance for the efficient execution of applications in large-scale grid environments. For instance, the work in [1] evaluates only the precision of estimation. In [2], only summarized results on convergence time and intrusion aspects were provided, which does not allow any further generalization of findings. In contrast, our work presents the metrics considered as a function of relevant parameters in diverse grid scenarios. Moreover, we evaluate the simultaneous use of the estimation tools, which is not considered in previous investigations.

1.3 Structure of the paper

The remainder of the paper is organized as follows. Section 2 briefly overviews bandwidth estimation procedures and introduces the `pathload` and the `abget` tools. Section 3 describes the experiments performed as well as discusses the results obtained. Section 4 draws some conclusions.

2 Tools for available bandwidth estimation in grid networks

Estimating the available bandwidth includes measuring different metrics [20] 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. Figure 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. In this paper, we focus on tools that estimate end-to-end available bandwidth.

In [19], a list of available bandwidth estimators is provided. Among them, `iperf` [21] is very popular among network administrators. `iperf` estimates the available bandwidth along a path by saturating the path with data sent between the two end points and measuring the amount of data sent. However, `iperf` is too intrusive and estimations are given as mean values. The intrusion issue can be ameliorated by employing TCP to send data between the end points and to start collecting measures after TCP slow start phase [22]. Another popular estimator is the `DIChirp` [23] which has access to the network card of a host to exchange packets between the host and another peer host. If the receiver detects a delay increase, the sending rate previous to the delay increase is considered to be the estimation of the available bandwidth. As the `iperf`, `DIChirp` provides estimations as mean values.

The `pathload` and `abget` tools perform their estimations using the Self-Loading Periodic Streams (SLoPS) technique. In this technique, several sequences of packets are transmitted between two end nodes using different rates, for measur-

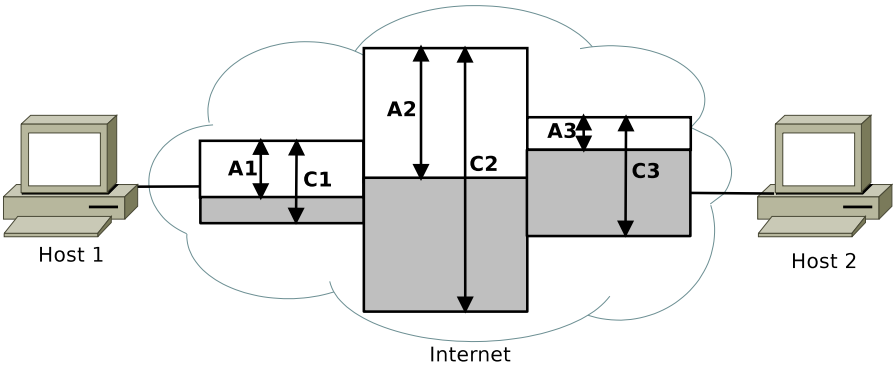


Fig. 1 Illustration of bandwidth metrics (based on a figure from [20])

ing the One-Way Delay (OWD)[1] of every packet sent at different rates. If an increase of the OWD value 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 the 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. `pathload` and `abget` estimate the end-to-end available bandwidth as described next.

2.1 `pathload`

To estimate the available bandwidth between hosts A and B using `pathload`, there is a need to execute a "sender" process at the host A and a "receiver" process at the host B. Information concerning the variations of the OWD value is exchanged between the 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 of no increase of OWD value, 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 the host A and the host B.

2.2 `abget`

To estimate the available bandwidth between the host A and the host B, an `abget` client at the host A should be directed to a TCP server (e.g. a web server) running either at the host B or at a host located in the same network where the 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 the ACKs sent by the host B. TCP segments with length equal to 1 MSS are sent to the host B. Upon receiving each segment, the host B sends one ACK to host A. If the host A sends segments in intervals of duration T (Rate $R = MSS/T$), the `abget` client induces the host B to send ACKs with a constant rate. Upon the arrival of ACKs at the host A, the OWD value is measured and the rate at which segments are generated is adjusted in order to find the interval $[R^{\min}, R^{\max}]$ which corresponds to the range of available bandwidth along the path between the two hosts. `abget` can employ two different types of search for the available bandwidth value: a binary search and a linear search. The binary search doubles the sending rate if it is lower than the available bandwidth and reduces it to half of it in case it is greater than the available bandwidth. In the linear search, the sending rate is increased/decreased by one unit of the sending rate.

[1] To measure OWD, the end nodes have to be synchronized. Possible solutions are the use of NTP (Network Time Protocol) servers, the adoption of GPS cards at both ends, or a software clock to enhance measurement accuracy without using GPS cards, such as the one proposed in [24].

Table 1 Comparing `abget` and `pathload`

Tool	Access to both hosts	Administrator Privileges	Firewall configuration
`pathload`	Necessary	Not necessary	Necessary
`abget`	Not necessary	Necessary	Not necessary

2.3 Differences between `pathload` and `abget`

The `pathload` and the `abget` differ in the way they implement the SLoPS technique. `pathload` requires processes to be executed at 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) to be active 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 using information related to the congestion control algorithms of TCP.

If, on the one hand, the main advantage of `abget` is to provide estimations by running a process only at one end host, on the other hand, it requires administrative privilege to operate TCP differently than the standard implementation in the local operating system. Another drawback is the need to inform manually several parameters to allow a relatively fast convergence to the interval corresponding to the available bandwidth estimation. `pathload` does not need these parameter values due to the employment of a TCP connection which allows the fine tuning of SLoPS in execution 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.

Table 1 compares the requirement of these tools.

3 Performance analysis

Several metrics were employed in the comparison; they were: accuracy of estimation, convergence time, level of intrusion in grid links, and the ability to handle simultaneously multiple estimations. Convergence time impacts the makespan of applications and it should be relatively short compared to the expected makespan of the application since the time to produce a task schedule should be short [12]. The level of intrusion in grid links impacts resource availability given that communication links are shared resources. Moreover, analyzing the ability to perform simultaneous estimations is quite relevant since in operational grids users and applications have asynchronous behavior.

Four different scenarios were employed to evaluate the performance of the `pathload` and the `abget` tools. The first scenario involves links of small nominal capacities (10 Mbps) and the second scenario links with large nominal capacities (1 Gbps). The third and the fourth scenarios extend the first and the second ones

by adding links and nodes. It is important to analyze these tools in all these scenarios due to the heterogeneity of the link capacities interconnecting clusters and of the number of clusters in a typical grid.

3.1 First scenario: links with small nominal capacities

The first scenario, illustrated in Fig. 2, was emulated using the NCTUns emulator [25] which allows the integration of simulated network topologies with real hosts running actual applications without requiring any modification of these applications. Estimations of the 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. Although a grid network is composed of numerous nodes and links, the information needed to evaluate the time for data transfer between a pair of nodes is the available bandwidth along the path connecting these two nodes.

In order to evaluate the tools under different network conditions, two virtual hosts were used in NCTUns to generate interfering traffic between the real hosts. Table 2 summarizes the configuration of the hosts involved in the experiments. The addition of interfering traffic to the traffic of a path for measuring the available bandwidth of a channel with well-known capacity is widely used in the literature ([1, 2, 9, 18]).

The topology used in Fig. 2 is an H-hop type. In such topology, there are H links between the source node and the destination node and the $((H+1)/2)$th link,

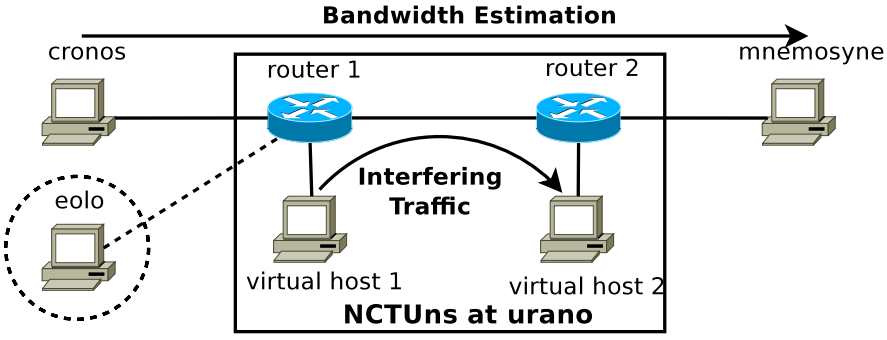


Fig. 2 Experimental setup with NCTUns

Table 2 Characteristics of the hosts used in the experiments

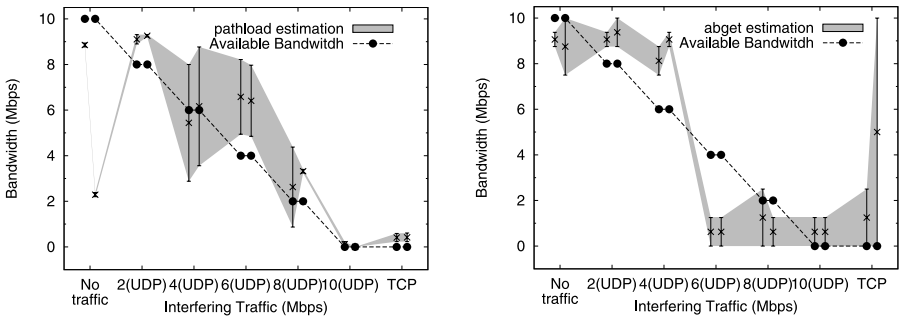
Host	Processor/Memory	Operating System (Linux)
eolo	Intel Core 2 Quad 2.66 GHz / 4 GB	Debian kernel 2.6.23.1
cronos	Intel Core 2 Quad 2.40 GHz / 4 GB	Debian kernel 2.6.23.1
mnemosyne	Dual Xeon 2 GHz / 4 GB	Debian kernel 2.6.23.1
urano	Intel Core 2 Duo 2.13 GHz / 4 GB	Fedora kernel 2.6.25.9

the so-called tight link, is the one which defines the available bandwidth along the path. This topology has been used in several other investigations ([9] and [1]). The topology in Fig. 2 emulates five clusters (or administrative domains), which is the same number of clusters used in [4] and in experiments using testbeds [16]. Actually, 71% of existing grids involve 1 to 10 domains. In the emulated scenario, each host can be seen as representing a cluster of machines and the links between them the links between hosts in these clusters whose available bandwidth needs to be estimated for the scheduling of tasks.

For each tool, three metrics were evaluated in the experiments: accuracy, execution time, and level of intrusion. The first scenario involved CBR UDP traffic sent at 2, 4, 6, 8, and 10 Mbps rates, TCP traffic, and no interfering traffic. The nominal capacity of the links was fixed on 10 Mbps during the measurement procedure. Two sets of measures were collected. In the first set, measures were collected between the hosts cronos and mnemosyne. Measures in the second set were collected simultaneously between the host cronos and the host mnemosyne, and between the host eolo and the host mnemosyne. Only the most relevant results are presented and discussed in this paper. In all experiments, the `abget` was configured to use the binary search since the linear search can take thrice longer to produce the desired estimation.

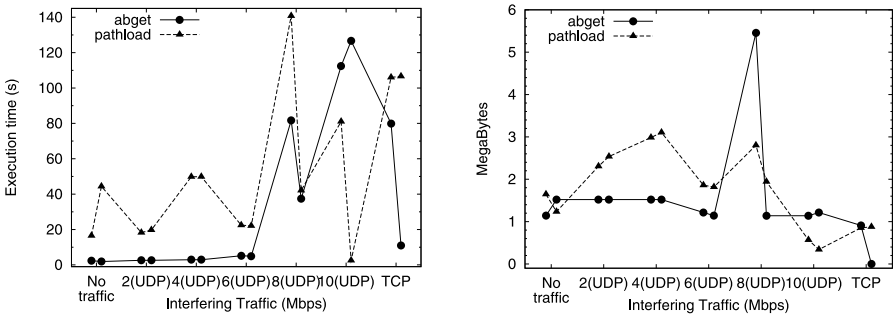
A relevant setting in the experiments was the disabling of Interrupt Coalescence (IC) in the network cards. IC is a feature of some network cards to decrease the amount of interruptions generated by the operating system when packets are either received or sent. When IC is enabled, interrupts are generated only after the existence of a certain amount of packets or at periodic intervals. As shown in [26], the precision of estimations of the available bandwidth are compromised when IC is enabled due to the extra delay the adoption of the IC feature incurs. In this way, we disabled IC in all experiments.

Figures 3 and 4 present the results obtained for the first set of measures. Figures 3(a) and 3(b) present results provided by `pathload` and `abget`, respectively, as a function of the interfering traffic. In these figures, the curve named "Available Bandwidth" shows 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 esti-



(a) `pathload` (b) `abget`

Fig. 3 Estimations provided by `pathload` and `abget` (first scenario)



(a) Execution time (b) Level of intrusion

Fig. 4 Performance of the estimation tools in the first scenario

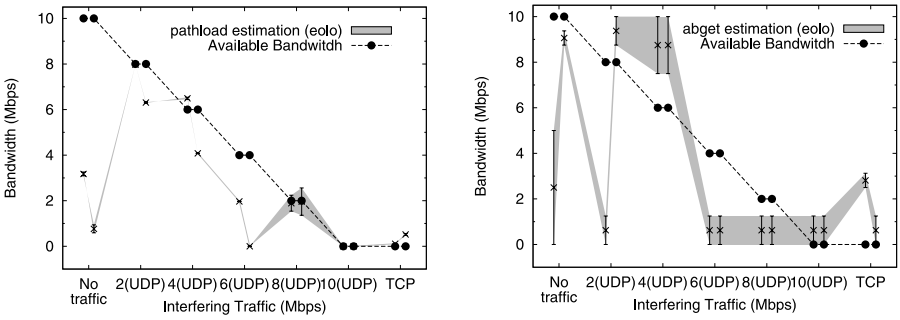
mation tools were executed twice. Results evince that `pathload` estimations were closer to the actual values when there was interfering traffic. Beside that, `abget` estimations did not follow the bandwidth availability along the path between cronos and mnemosyne. With interfering traffic less or equal to 4 Mbps, `abget` estimated that the path was almost 100% available, whereas with interfering traffic greater or equal to 6 Mbps estimations indicated that the link was unavailable. The estimation intervals from `abget` were on average larger under TCP interfering traffic than when they were under other interfering traffic.

The execution time of the estimation tools is shown in Fig. 4(a). Since `abget` does not demand an initial value of the transmission rate close to the nominal capacity, it requires the transfer of the same quantity of data at the beginning of the estimation regardless the availability of the links. As a consequence, as the links become more utilized, the execution time tends to increase, as can be observed in the "abget" curve. Nevertheless, when the interfering traffic is small ($\leq$8 Mbps), `abget` converges to results faster than does `pathload`. The low execution time of `abget` in one of the executions carried out with TCP interfering traffic was due to the fact that the tool was not able to connect to the web server at the host mnemosyne under heavy load.

The level of intrusion expressed as the volume of injected bytes by each tool is shown in Fig. 4(b). We observe that `abget` is more stable 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 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`.

Some of the results obtained in the second set of measures are shown in Figs. 5 and 6. Results for the levels of intrusion of the tools were quite similar to those obtained with just one instance of the estimation tools in execution, so they have been omitted here for the sake of clarity. We make a remark that 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.

Figures 5(a) and 5(b) show the results when `pathload` and `abget` estimated the available bandwidth between the host eolo and the host mnemosyne. Results consid-



(a) `pathload` estimations (b) `abget` estimations

Fig. 5 Simultaneous executions in the first scenario—eolo case

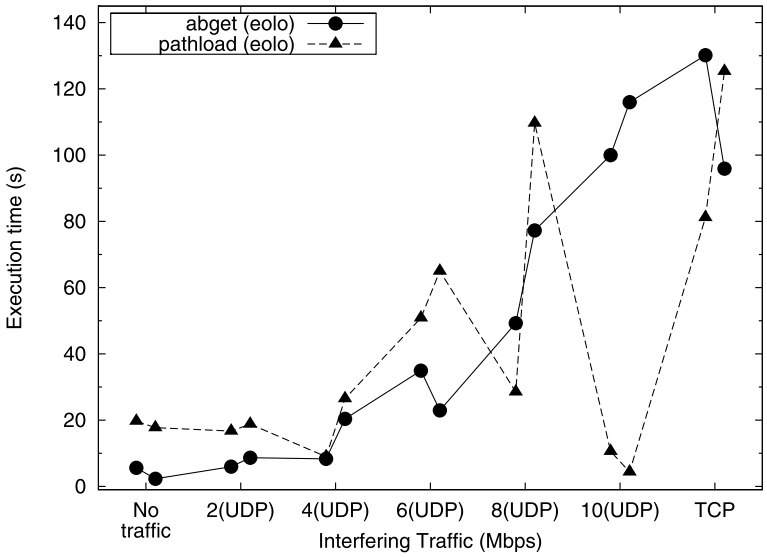
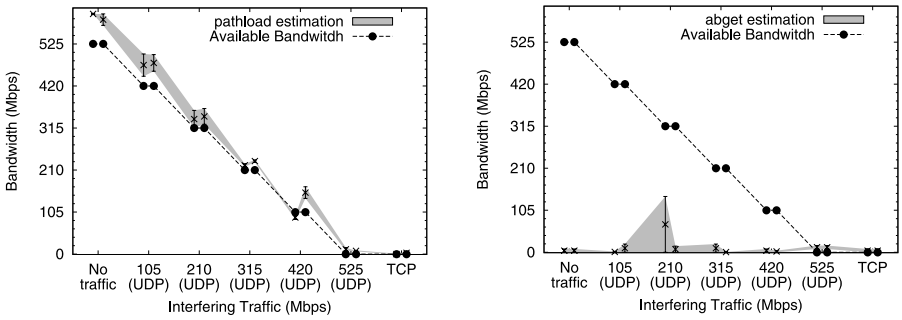


Fig. 6 Execution time of tools (simultaneous execution—eolo case)

ering the hosts cronos and mnemosyne were quite similar and they will not be shown. Again, `pathload` provided more accurate estimations than `abget`. The main difference between these results and those obtained in the first set of measures (see Fig. 3) is the width of intervals given by `pathload` in the second set. UDP traffic sent at fixed rate decreases the bandwidth availability and, as a consequence, intervals given by `pathload` decrease.

The execution time values provide the main difference between the experiments with two simultaneous executions and the experiments with a single execution. Figure 6 shows the execution time for samples collected between the host eolo and the host mnemosyne (similar results were observed between the host cronos and the host mnemosyne). Comparing the execution time when a single estimation was involved (Fig. 4(a)), an increase in execution time of the `abget` tool can be observed.



(a) `pathload` (b) `abget`

Fig. 7 Estimations provided by `pathload` and `abget` (second scenario)

Such increase started at a rate of 4 Mbps (UDP) while it started at a rate of 8 Mbps (UDP) when only one execution was involved.

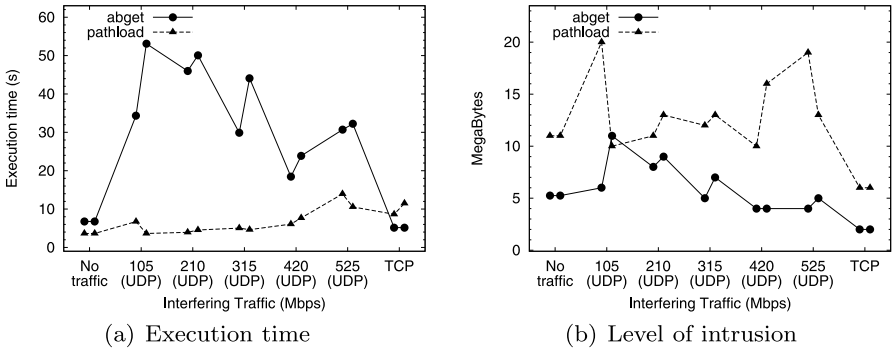
In general, in the first scenario, `pathload` provided better estimations than `abget`, while `abget` executed faster and was less intrusive. Both tools executed relatively fast (<2 m 20 s) and with a relatively low level of intrusion (<6 MB), considering the expected amount of data transfer done by typical grid applications.

Results cannot be compared to those in [1] and in [2] since none of these works considered 10 Mbps links.

3.2 Second scenario: links with large nominal capacities

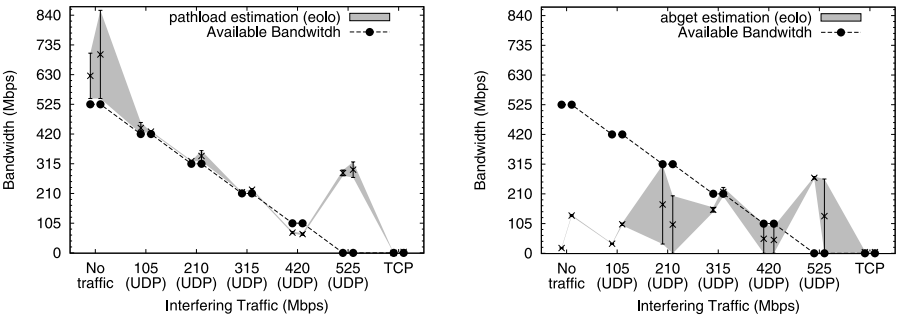
The second scenario includes a local network with links of 1 Gbps interconnecting the hosts; `NCTUns` is not used in this scenario. The `iperf` [21] tool was employed to generate interfering traffic. Virtual hosts and routers were created in the urano machine (see Fig. 2) using virtual network interfaces. Estimations were performed using the same hosts of the first scenario. Although links have nominal capacity of 1 Gbps, the observed actual capacity was 525 Mbps, so this value was considered as the reference value to the maximum available capacity. The same metrics and steps of the first scenario were considered in this second scenario. The difference between these two scenarios is related to the rates of UDP interfering traffic, which are: 105, 210, 315, 420, and 525 Mbps.

Figure 7 shows the accuracy of the tools when estimations were performed only between the hosts cronos and mnemosyne. Estimations provided by `pathload` (Fig. 7(a)) were more accurate than those given by `abget` (Fig. 7(b)). In contrast with the first scenario, `pathload` estimations clearly track the available bandwidth value. Similarly to the first scenario, `pathload` provided more accurate estimations under higher levels of interfering traffic. Beside that, `abget` presented a different behavior when compared with that observed in the first scenario. Although the grid links had larger nominal capacity, `abget` estimated that links were almost 100% occupied regardless of the intensity of interfering traffic. Although the tuning of parameters in `abget` was carried out, results did not improve. Actually, the need to set a high number of parameters in `abget` is a major shortcoming, specially when under highly dynamic environments such as grid networks.



(a) Execution time (b) Level of intrusion

Fig. 8 Performance of the estimation tools in the second scenario



(a) `pathload` estimations (b) `abget` estimations

Fig. 9 Simultaneous executions in the second scenario—eolo case

The execution time of `abget` was on average much higher than that of `pathload`, as shown in Fig. 8(a). The `pathload` tool took longer periods to execute than `abget` did under the influence of TCP interfering traffic. `pathload` migrates to the next sending rate when it detects instability of the delay values of the first packets sent at a certain rate, while `abget` always sends the same number of packets at all rates. As a consequence, `abget` took much longer due to the larger number of rates to cover.

Figure 8(b) shows that `pathload` injected more traffic than `abget`. Moreover, the variability of bytes sent by `abget` was lower than that of `pathload`. Comparing these results with those given in Fig. 4(b), it can be observed that the level of intrusion increased with the increase of the link capacity and that the level of intrusion had a larger variability when interfering traffic was composed of UDP traffic. However, when TCP interfering traffic was used, both estimation tools produced more stable results.

Figures 9 to 11 show the results when two simultaneous estimations were pursued. Intervals produced by `abget` were wider (Fig. 9) and the `pathload` tool produced estimations that diverged from the real availability when links were congested. The

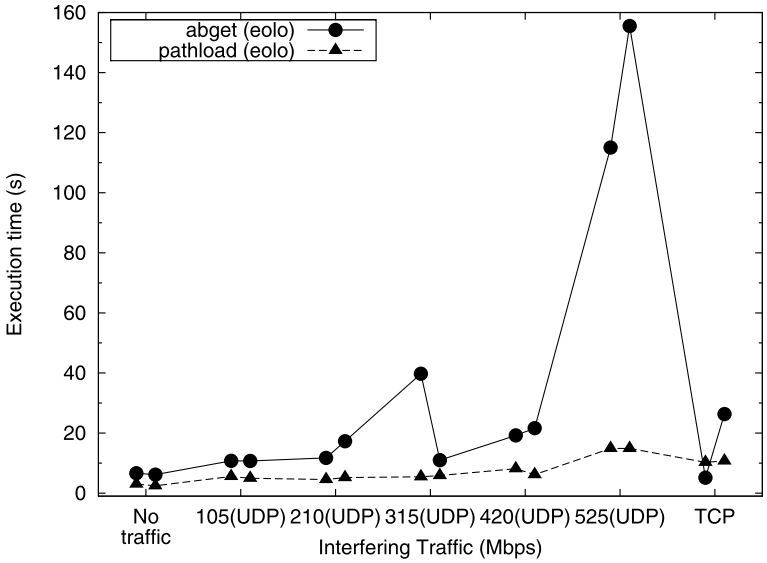


Fig. 10 Execution time of the estimation tools (simultaneous executions—eolo)

enlargement of the intervals provided by `pathload` also happened when no interfering traffic existed.

Another point to highlight is the significant increase in the execution time of `abget` in congested network. The execution time of `abget` increased by one order of magnitude when UDP interfering traffic grew from 420 Mbps to 525 Mbps (Fig. 10).

It is important to mention the behavior of `pathload` in the second set of measures, specially when the transmission rate of the UDP interfering traffic was 105 Mbps (Fig. 11). In one of the executions, this tool generated about 300 MB of data, a volume far from being negligible. Such behavior evinces that available bandwidth estimation tools can be highly intrusive, which motivates the implementation of procedures to avoid overhead. A simple solution would be either to request users an estimation of an upperbound for the execution time or to provide an upperbound for the injected amount of bytes. Although external criteria for stopping the execution of the estimators may result in less precise estimations, the overall outcome can be preferable if all the metrics are jointly evaluated.

In summary, for the second scenario, `pathload` provided good results even when used in scenarios with high capacity links. The execution time of `pathload` was lower than that of `abget`, although `abget` was less intrusive on the average. Schedulers using `abget` to estimate available bandwidth would provide schedules that would lead to underutilization of network links due to the underestimation of the available bandwidth. Moreover, the `pathload` tool has the potential to flood the network with traffic that may directly impact the execution of other applications.

Results involving pathload and UDP traffic are in line with those reported in [1] and [2]. The other results in these two references cannot be compared to those of our experiments due to the differences in the scenarios used.

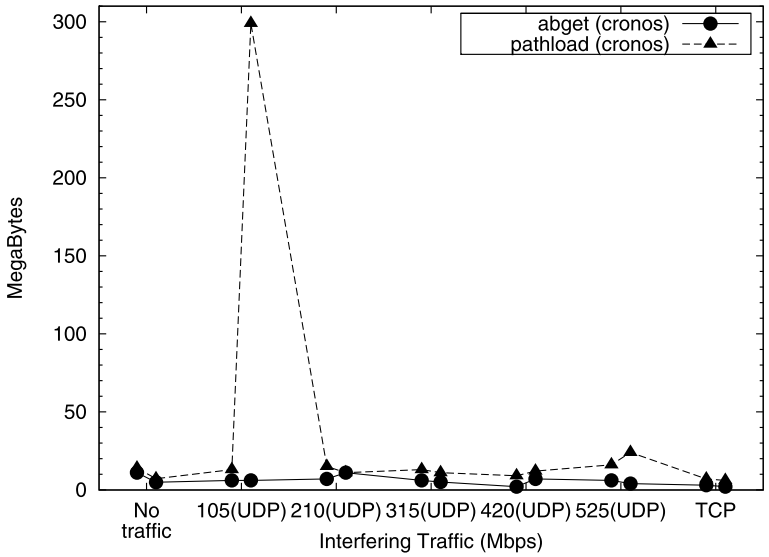


Fig. 11 Level of intrusion of the estimation tools (simultaneous executions—cronos)

3.3 Third scenario: links with small nominal capacities and $H = 3$

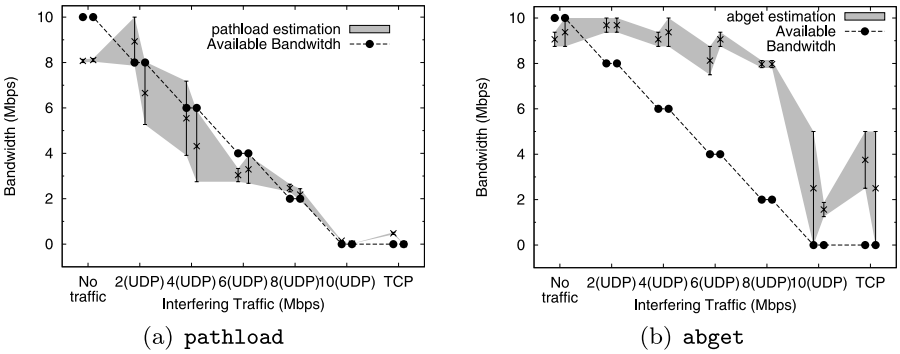
In order to evaluate the tools in large networks, experiments were conducted with H-hop type topologies for $H = 3$ and $H = 5$. Due to space limitation only results with $H = 3$ are reported in the paper. Results for $H = 5$ were consistent with those for $H = 3$.

An H-hop network with $H = 3$ corresponds to a network with 9 clusters and 14 links connecting them, which is an increase of 80% and 133% in the number of clusters and links comparing to the previous examples. An H-hop type network with $H = 3$ is quite realist since the number of clusters in grids typically varies from five to 10.

Figures 12(a) and 12(b) show the precision of estimations as a function of the load for the `pathload` and for the `abget`, respectively. As in the experiments with a network with $H = 1$, the most precise results given by `pathload` were under interference traffic. The `abget` tool showed the usual performance: when UDP interfering traffic intensity is lower than 10 Mbps, it estimates 100% availability; however, whenever the UDP interfering traffic is higher than 10 Mbps, it estimates that the link is 100% utilized. Similar results than those when $H = 1$ were found since the increase in the number of hops impacts the end-to-end delay but not the difference between end-to-end delays.

Since `abget` transfers the same amount of bytes, its execution time increases with the increase of the load (Fig. 13) as usual. It can be noted in Fig. 13 that the execution time increases when the interfering traffic is higher than 6 Mbps.

Results with $H = 3$ reinforce those with $H = 1$: the `pathload` is quite more precise under low loads while `abget` underestimate results under this traffic condition. However, `pathload` demands more time to produce results.



(a) `pathload` (b) `abget`

Fig. 12 Estimations provided by `pathload` and `abget` (third scenario)

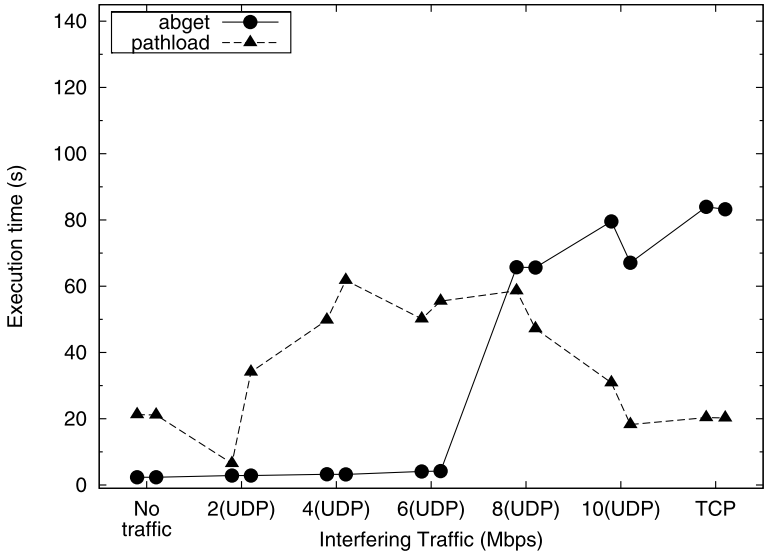


Fig. 13 Execution time of the estimation tools in the third scenario

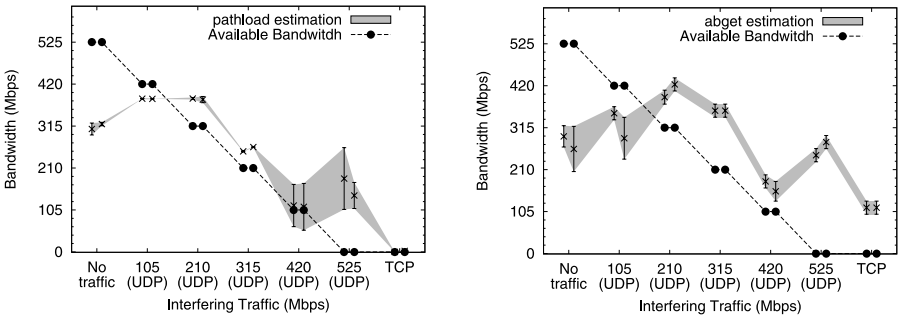
3.4 Fourth scenario: links with large nominal capacities and $H = 3$

Experiments with $H = 3$ and links with 1 Gbps reinforce that `pathload` provides much more precise estimations (Fig. 14) than `abget`, although the increase in the number of links along a path made `pathload` to overestimate the available bandwidth.

The increase in the number of links speeded up `abget` computation (Fig. 15(a)). However, overall, `pathload` clearly outperform `abget`.

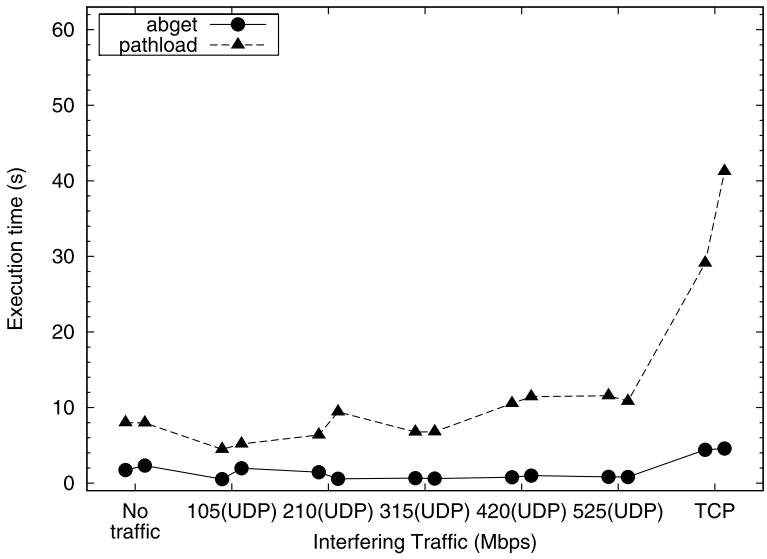
3.5 Summary of results obtained

Table 3 summarizes results found in the experiments. It can be noticed that the performance of `pathload` was quite satisfactory except for simultaneous estimations



(a) `pathload` (b) `abget`

Fig. 14 Estimations provided by `pathload` and `abget` (fourth scenario)



(a) Execution time

Fig. 15 Performance of the estimation tools in the fourth scenario

and when using links with 1 Gbps capacity. As the scenarios become more complex, the execution time of `abget` becomes lower than that of `pathload`. Moreover, `pathload` always injects more traffic in the network. Another disadvantage is that `pathload` requires code modification for simultaneous estimations of the available bandwidth.

Beside that, the simultaneous use of various processes in the same host requires code modification of `pathload` which is a drawback of this tool.

Table 3 Summary of the qualitative comparison

Metric	`pathload`	`abget`
Estimations (single execution, 10 Mbps)	more precise	less precise
Estimations (simultaneous executions, 10 Mbps)	more precise	less precise
Execution time (single execution, 10 Mbps)	High	Low
Execution time (simultaneous execution, 10 Mbps)	Similar	Similar
Intrusion (single execution, 10 Mbps)	High	Low
Intrusion (simultaneous executions, 10 Mbps)	High	Low
Estimations (single execution, 1 Gbps)	more precise	less precise
Estimations (simultaneous executions, 1 Gbps)	more precise at high availability	less precise
Execution time (single execution, 1 Gbps)	Low	High
Execution time (simultaneous execution, 1 Gbps)	Low	High
Intrusion (single execution, 1 Gbps)	High	Low
Intrusion (simultaneous executions, 1 Gbps)	High	Low

4 Conclusion and future work

Resource availability in grids is not only a function of its heterogeneous and decentralized nature but also of the variable level of concurrency found in grid networks. Having a network-based high performance computing environment only renders this scenario more challenging. Estimating the available bandwidth in network links thus plays a key role for scheduling and rescheduling of tasks, specially to highly demanding e-Science applications which are strongly dependent on the transfer of large amount of data.

This paper evaluated the performance of two tools for the estimation of the available bandwidth in network scenarios. Our performance analysis has been driven by the potential integration of these tools into grid systems for helping the decision making process of (re)scheduling the tasks of applications. Several metrics were considered such as estimation accuracy, execution time, level of intrusion, and effectiveness when concurrent estimations were performed.

Overall, `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 can have in other flows during measurements. Furthermore, `pathload` has to be executed in both end hosts, a negative characteristic that might motivate modifications in `abget` to improve its accuracy. The code of `pathload` must also be changed so that the adopted ports can be dynamically defined, allowing the simultaneous use by several pairs of end hosts in large-scale distributed grid environments.

We suggest to replicate the investigation using larger environments such as those available in PlanetLab [27].

References

1. Jain M, Prasad RS, Dovrolis C (2003) Evaluating pathrate and pathload with realistic cross-traffic. Talk at bandwidth estimation workshop 2003. http://www.caida.org/workshops/isma/0312/slides/rprasad-best.pdf. Retrieved August 17, 2009
2. Shriram A, Murray M, Hyun Y, Brownlee N, Broido A (2005) Comparison of public end-to-end bandwidth estimation tools on high-speed links. In: Passive and active network measurement: 6th international workshop, PAM 2005. Springer, Berlin
3. Foster I (2002) What is the grid? A three point checklist. GRID Today 1(6). http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf. Retrieved August 17, 2009
4. El Khatib Y, Edwards C (2007) A survey-based study of grid traffic. In: GridNets '07: Proceedings of the first international conference on networks for grid applications. ICST, Brussels, pp 1–8
5. Newman HB (2004) Networking for high energy and nuclear physics as global e-science. In: Proceedings of computing in high energy and nuclear physics, Sep 2004. http://ultralight.caltech.edu/web-site/common/publications/article/2004/09chep04/Network_for_HEP.pdf. Retrieved August 17, 2009
6. Ferrari T, Giacomini F (2004) Network monitoring for GRID performance optimization. Comput Commun 27(14):1357–1363. Special Issue on Network Support for Grid Computing
7. Silvester JA (2005) CalREN: Advanced network(s) for education in California, Oct 2005. http://isd.usc.edu/~jsilvest/talks-dir/20051021-cudi-merida.pdf. Retrieved August 17, 2009
8. WLCG Worldwide LHC Computing Grid (2008) http://lcg.web.cern.ch/LCG/public/. Retrieved August 17, 2009
9. Jain M, Dovrolis C (2003) End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Trans Netw (TON) 11(4):537–549
10. Batista DM, Drummond AC, Fonseca NLS (2009) Robust scheduler for grid networks. In: SAC '09: Proceedings of the 2009 ACM symposium on applied computing, New York, NY, USA, Mar 2009. ACM Press, New York, pp 35–39
11. Ito T, Ohsaki H, Imase M (2005) On parameter tuning of data transfer protocol GridFTP for wide-area grid computing. In: Proceedings of the BroadNets 2005, vol 2, pp 1338–1344
12. Batista DM, da Fonseca NLS, Miyazawa FK, Granelli F (2008) Self-adjustment of resource allocation for grid applications. Comput Netw 52(9):1762–1781
13. Huedo E, Montero RS, Llrorent IM (2002) An experimental framework for executing applications in dynamic grid environments. Technical Report 2002-43. NASA Langley Research Center
14. Montero RS, Huedo E, Llorente IM (2003) Grid resource selection for opportunistic job migration. In: Proceedings of the 9th international Euro-Par conference. Springer, Berlin, pp 366–373
15. Allen G, Angulo D, Foster I, Lanfermann G, Liu C, Radke T, Seidel E, Shalf J (2001) The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment. Int J High Perform Comput Appl 15(4):345–358
16. Chun G, Dail H, Casanova H, Snavely A (2004) Benchmark probes for grid assessment. In: Proceedings of the 18th international parallel and distributed processing symposium, Apr 2004, pp 276–283
17. Iosup A, Li H, Jan M, Anoep S, Dumitrescu C, Wolters L, Epema DHJ (2008) The grid workloads archive. Future Gener Comput Syst 24(7):672–686
18. Antoniades D, Athanatos M, Papadogiannakis A, Markatos EP, Dovrolis C (2006) Available bandwidth measurement as simple as running wget. In: Proceedings of the passive and active measurement workshop 2006
19. Cooperative Association for Internet Data Analysis (CAIDA) (2008) CAIDA : tools : taxonomy. http://www.caida.org/tools/taxonomy/performance.xml#bw. Retrieved August 17, 2009
20. Prasad R, Dovrolis C, Murray M, Claffy K (2003) Bandwidth estimation: Metrics, measurement techniques, and tools. IEEE Netw 17(6):27–35
21. NLANR (2008) Iperf. http://sourceforge.net/projects/iperf/?abmode=1. Retrieved August 17, 2009
22. Tirumala A, Cottrell L, Dunigan T (2003) Measuring end-to-end bandwidth with Iperf using Web100. Technical Report SLAC-PUB-9733, Stanford Linear Accelerator Center
23. Ozturk Y, Kulkarni M (2008) DIChirp: Direct injection bandwidth estimation. Int J Netw Manag 18(5):377–394
24. Pásztor A, Veitch D (2002) PC-based precision timing without GPS. In: ACM, SIGMETRICS, Los Angeles, CA, USA, June 2002
25. Wang SY, Choua CL, Lin CC (2007) The design and implementation of the NCTUns network simulation engine. Simul Model Pract Theory 15(1):57–81
26. Prasad R, Jain M, Dovrolis C (2004) Effects of interrupt coalescence on network measurements. In: Proceedings of the passive and active network measurement workshop 2004, pp 247–256

27. PlanetLab (2009) An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/. Retrieved August 17, 2009

Daniel M. Batista received a B.Sc. degree in Computer Science from Federal University of Bahia in 2004 and his M.Sc. degree in Computer Science from State University of Campinas in June 2006. He is now a Ph.D. candidate at the Institute of Computing, State University of Campinas, Campinas, Brazil and he is affiliated with the Computer Networks Laboratory at the same university. His research interests include traffic engineering and grid networks. His current research addresses robust mechanisms for grid networks.

Luciano J. Chaves was born in the state of Minas Gerais, Brazil. He received his B.Sc. in Computer Science (with emphasis in Computer Networking and Distributed Systems), from the Federal University of Juiz de Fora, in 2007. He is now a M.Sc. student at the Institute of Computing of the University of Campinas, Campinas, Brazil. He is affiliated with the Computer Networks Laboratory at the same university, and his research interests include wireless networks, cognitive networks and autonomic computing. His current research addresses cognitive algorithms for rate adaptation in wireless networks.

Nelson L.S. da Fonseca received his Electrical Engineer (1984) and M.Sc. in Computer Science (1987) degrees from The Pontificial Catholic University at Rio de Janeiro, Brazil, and the M.Sc. (1993) and Ph.D. (1994) degrees in Computer Engineering from The University of Southern California, U.S.A. Since 1995, he has been affiliated with the Institute of Computing of The State University of Campinas, Campinas—Brazil where he is currently a Full Professor. He is also a Consulting Professor to the Department of Informatics and Telecommunications of the University of Trento, Italy. He is the Editor-in-Chief of the IEEE Communications Surveys and Tutorials. He served as Editor-in-Chief of the IEEE Communications Society Electronic Newsletter and Editor of the Global Communications Newsletter. He is a member of the editorial board of: Computer Networks, IEEE Communications Magazine, Peer-to-Peer Networking and Applications and International Journal of Communication Systems. He served on the editorial board of the IEEE Transactions on Multimedia, Brazilian Journal of Computer Science, and on the board of the Brazilian Journal on Telecommunications. He is the recipient of Elsevier Computer Networks Editor of the Year 2001, USC International Book Award and of the Brazilian Computing Society First Thesis and Dissertations Award. He is an active member of the IEEE Communications Society. He is currently ComSoc Director for Latin America. He served as ComSoc Director of On-line Services (2002–2003) and served as technical chair for several ComSoc symposia and workshops.

Artur Ziviani was born in Rio de Janeiro, Brazil. He received the B.Sc. degree in Electronics Engineering in 1998 and the M.Sc. degree in Electrical Engineering (with emphasis in Computer Networking) in 1999, both from the Federal University of Rio de Janeiro (UFRJ), Brazil. In December 2003, he received the Ph.D. degree in Computer Science from the University Pierre et Marie Curie (Paris 6), Paris, France, where he has also been a lecturer during the 2003–2004 academic year. Since September 2004, he is with the National Laboratory for Scientific Computing (LNCC), located in Petrópolis, Brazil, where he heads with Antonio Tadeu Azevedo Gomes the MARTIN R&D group. From September 2008 to January 2009, he was a visiting researcher at the INRIA's ASAP and A4RES teams in France. He is a member of SBC (the Brazilian Computing Society), IEEE, and ACM.