

Accurate Autonomous Accounting in Peer-to-Peer Grids

Robson Santos
robson@dsc.ufcg.edu.br

Francisco Brasileiro
fubica@dsc.ufcg.edu.br

Alisson Andrade
aandrade@dsc.ufcg.edu.br

Nazareno Andrade
nazareno@dsc.ufcg.edu.br

Walfredo Cirne
walfredo@dsc.ufcg.edu.br

Universidade Federal de Campina Grande
58.109-970, Campina Grande, PB, Brasil
Phone: +558333101365

ABSTRACT

We here present and evaluate an autonomous accounting scheme that provides accurate results even when the parties (consumer and provider) do not trust each other. Our accounting scheme relies on the observed relative performance among the parties. It is totally autonomous in the sense that it uses only local information, i.e. there is no exchange of information between the parties. This allows for the deployment of the autonomous accounting without requiring any sort of identification infrastructure, such as certificate authorities. The no need of trust or sophisticated infrastructure make our accounting scheme a perfect fit for peer-to-peer grids, which aim to scale much further than traditional grids by allowing free unidentified entry into the grid. Our results show that the proposed scheme performs very close to a perfect accounting scheme whose implementation is infeasible in most systems, including those we target. Although our autonomous accounting scheme was developed to work with OurGrid, it can also be useful for other systems. The basic requirement to use our accounting scheme is that resource consumers must also be resource providers.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Grid Computing

General Terms

Measurement, economics, performance, experimentation.

Keywords

Accounting, resource usage, reputation model, peer-to-peer grids.

1. INTRODUCTION

Computational grids have appeared about a decade ago with the promise of delivering great amounts of computational power to parallel applications by enabling the seamless sharing of resources spread among different administrative domains. The grid promise has latter evolved into enabling on-demand access and composition of computational services located throughout the globe. Meanwhile, the first grid systems went into production: TeraGrid, CERN's LCG, OurGrid, Grid3, and many others. This is not to say, however, that all grid problems have been solved. We are still far from that. In particular, we would like to draw the reader's

attention to *accounting* and *scalability*; two issues in grid computing that clearly demand further research.

Accounting aims to measure resource usage. A grid provider typically wants to gauge how much resources a given client consumed, either to control such usage or to charge for it. Since a grid comprises multiple administrative domains, this raises a question of trust: should the consumer believe the accounting gauged by the provider? Moreover, finding a metric to represent resource usage is not a simple task, as we can see in Section 2.

Scalability is also an issue because grids today require complex installation and set-up, and, more importantly, demand human negotiation among the different administrative domains to define how resources are going to be accessed and shared. This is the case because grids aim to provide sophisticated services, typically with some guarantee on what users can expect of such services [11]. Using peer-to-peer techniques has appeared as a promising approach to foster grid scalability [11]. In fact, there are now many peer-to-peer grid projects, e.g. [6] [7] [15] [17], and the first peer-to-peer grids are also going into production. Naturally, peer-to-peer grids do not yet provide services as rich as traditional grids. Application execution (i.e. CPU sharing) appears to be the first service being tackled in peer-to-peer grids. On the other hand, peer-to-peer grids aim to scale for the planet! They try to create a global scale resource-sharing network, so that anyone who requires large amounts of computing power needs just to connect to this network and use the resources that are available.

But notice that accounting becomes an even tougher problem for peer-to-peer grids. In traditional grids, where participants are known and authenticated, there is typically reasonable trust among the administrative domains, and accounting solutions can rely on such trust. On a peer-to-peer grid, on the other hand, participants can freely join the system, without providing a strong link to their real meat-space identities [8]. Therefore, participants are essentially anonymous, and no solutions (including accounting) can be built assuming trust among the parties.

Yet, accounting may be even more important for peer-to-peer grids than it is for traditional ones. This is because free-riding is a common behavior in peer-to-peer systems [1] [13] [16]. A free-rider is a peer that only consumes resources, never contributing back to the system. If there is a non-zero cost for providing resources to the grid, and free-riders get the same benefit from the system as peers that contribute to the system, then it is in the best interest of peers to free-ride and the system may collapse [4]. Coping with free-riding in a peer-to-peer system typically involves some reciprocation strategy, on which a given peer tries to help peers that have helped it in the past [2] [9]. Therefore, it is crucial for a peer to be able to gauge the contribution another peer provided to it, a requirement that, in a CPU sharing peer-to-peer grid, translates into a strong need for accurate accounting.

"Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MGC '05, November 28- December 2, 2005 Grenoble, France Copy-right 2005 ACM 1-59593-269-0/05/11... \$5.00"

We here present an autonomous accounting scheme that relies on the relative compute power of each peer to provide accurate accounting for peer-to-peer grids. We rely on the fact that, in a peer-to-peer grid, resource consumers are also resource providers to autonomously evaluate the relative performance of the parties, and then use this information to normalize the time a resource has been allocated by a provider to a consumer.

Moreover, our accounting scheme is totally autonomous in the sense that it can be fully evaluated using only local information. In fact, the best and simplest way to avoid the need of trust relationship among the parties in the system is autonomously accounting the resource allocations. Our autonomous accounting system does not need a complex infra-structure for its deployment as cryptographed certificates, trusted auditor peers, banking or secure e-cash because no accounting information is exchanged among the participants. Moreover, the accounting values are independently maintained by each peer locally.

The rest of this paper is structured as follows. In Section 2 we describe the accounting problem and discuss some possible approaches to solve it, including the definition of both a perfect and a simplified time-based accounting scheme. We also discuss the difficulties and drawbacks of using these schemes. Section 3 presents our autonomous scheme that uses the relative computational power of peers to accurately account resource utilization. Then, in Section 4, we analyze how well our autonomous accounting performs, comparing it against the perfect and the time-based schemes. In Section 5 we discuss related works. Section 6 concludes the paper with our final remarks.

2. THE RESOURCE ACCOUNTING PROBLEM

The accounting of a resource allocation is a function of (i) the processing power that the computing resource provided (or used, depending on the viewpoint of who is calculating the allocation's value) is able to deliver for a particular task at a given point in time, and of (ii) the time interval during which the resource is provided (or used). Assuming that $power_p(\tau, t)$ is the amount of processing power a computing resource P is able to deliver to a task τ at a particular instant of time t , the *perfect accounting* of running task τ on processor P starting at some time t_i , for the inter-

val of time Δt would be $AP(\tau, P) = \int_{t_i}^{t_i + \Delta t} power_P(\tau, t) dt$.

As can be seen, the difficulty in providing accurate accounting lies in determining $power_p(\tau, t)$ for each instant of time t . Assessing $power_p(\tau, t)$ is complicated by two main factors. First, due to inherent characteristics of the tasks that are executed in the grid, the same computing resource may deliver different power to different tasks even when the resource is subject to the same competing load and allocated for the same amount of time. For instance, a resource with a fast CPU but with a slow disk may be more valuable to CPU-intensive tasks than to I/O-intensive ones. Second, this information has to be inferred autonomously by the consumer peer, without relying on information of untrustworthy peers. In particular, the consumer cannot trust the $power_p(\tau, t)$ informed by the provider of P .

Consequently, it is unlikely that such a perfect accounting scheme can be built, unless some restrictions are imposed. Alternatively, one can try to develop practical accounting schemes that are

just good approximations to the perfect scheme. A simple approximation would be to consider that all computing resources have the same power at any given time. For instance, one can assume that all resources have power 1 at all times and simply use the time a resource has been made available to a remote peer as the accounting of its resource allocation. Using such time-based scheme, the accounting of running task τ on processor P starting at some time t_i , for the interval of time Δt would produce $AT(\tau, p) = \Delta t$.

Since the time interval a resource is made available is easily and autonomously computed by both consumer and provider, this approach can easily be deployed on grids in which peers do not trust each other. However, such a time-based accounting scheme does not take into account the amount of work done by the resource. Thus, peers whose resources take longer to process tasks will get more benefits from the grid than those whose resources are faster and execute the same tasks faster. In Section 4.4 we discuss the negative impacts that such an approach may bring to the grid as a whole.

3. ACCURATE AND AUTONOMOUS ACCOUNTING

In order to circumvent the problems just described, we have developed a new approach to do accounting that considers the relative power of the peers in the grid. In a peer-to-peer grid, a peer is typically responsible for a site (i.e. resources under a single administrative domain, normally connected by a LAN). Since reputation models normally rank peers rather than resources [2] [3], we make our accounting scheme calculate the accounting per peer.

A peer has local resources that are used to run some tasks of its jobs and, when idle, to run tasks from remote peers' jobs. We take advantage of this fact to define a new metric, named relative power $rpower$, which is used to account for task executions. The relative power of a remote peer is a measurement of how much faster or slower this peer is, when compared to the local peer. That is, $rpower_B(A)$ is the relative power that a peer A is currently able to deliver, as perceived by a peer B . It is estimated by peer B as the average execution time of the tasks B ran locally, divided by the average execution time of the tasks B ran on A .

That is, $rpower_B(A) = \frac{mean\{exectime(\tau_B B)\}}{mean\{exectime(\tau_B A)\}}$, where

$\{exectime(\tau_B B)\}$ denotes the execution time of the tasks that B ran on B (i.e. locally), whereas $\{exectime(\tau_B A)\}$ denotes the execution time of the tasks that B ran on A . Also, for any peer A , $rpower_A(A) = 1$.

The relative accounting of running task τ from peer B on peer A starting at some time t_i , for the interval of time Δt is defined in B as $AR_B(\tau, B, A) = \Delta t \times rpower_B(A)$ and in A as $AR_A(\tau, B, A) = \Delta t \times rpower_A(A)$. Note that, since $rpower_A(A) = 1$ for any peer A , when A is providing resources to another peer B , it accounts this allocation by simply measuring the amount of time each of its resources are made available to B .

Furthermore, as both the consumer and the provider can measure the time a resource has been in use as well as evaluate their power relative to that of the resource provider, our scheme is completely autonomous and does not require peers to share information. On the other hand, as we rely on the average execution

time of tasks, our scheme works better when tasks demand similar amounts of processing. Heterogeneity in execution time of tasks may introduce errors in our accounting. However, as we shall see in Section 4.4, this error is not significant in the representative settings we have analyzed.

4. PERFORMANCE EVALUATION²

We have used simulations to analyze the different accounting schemes. We wrote a new event-based simulator for the peer-to-peer computational grid described in Section 4.1 and implemented the three accounting discussed: perfect¹, time-based and relative accounting schemes.

4.1. Peer-to-Peer Grid Model

Typically, in a peer-to-peer computational grid, each peer represents an administrative domain. Each peer controls the access to a set of local computational resources and plays two roles: (i) it may work as a provider, allowing its (otherwise) idle resources to execute other peers' tasks, and (ii) it may work as a consumer, using its local resources and any idle resources from other peers to execute its own tasks. However, as pointed out before, free-riding is a common behavior in peer-to-peer systems [1] [16].

We thus assume that there is a mechanism to discourage free riding. More precisely, we assume that peers reciprocate resource allocations using the Network of Favors (NoF) [2] [3]. NoF is an efficient and lightweight reputation-based resource allocation mechanism designed to promote contributions of resources in a peer-to-peer grid [3]. In NoF, the more resources a peer provides, the more likely it is that it will receive resources from other peers when it needs them. It has been used to promote resource sharing in OurGrid [8].

In NoF terminology, resource allocation is called a *favor*. Resource providers give favors and resource consumers receive them. For a peer A , the reputation ranking of a peer B is represented by the balance of favors they have exchanged, i.e. the amount of favors A has received from B minus the amount of favors A has provided to B . Also, to prevent id-changing attacks², balances are always kept non-negative. When a peer has idle resources, it provides them to the grid. If two remote peers request these resources, then the remote peer with the highest local balance gets the resources. However, users of the local peer always have priority over those from remote peers. Thus, whenever tasks from local users are submitted, any tasks from remote users are aborted. This simple set of norms makes it in the best interest of peers to provide to the grid as much computational power as they can [3].

Note that NoF is completely decentralized and autonomous, in the sense that there is no exchange of reputation information among peers. In particular, for any 3 peers A , B and C , the reputation of C in A 's eyes bears no relation with the reputation of C in B 's eyes. The reputation ranking is calculated using only local information about the favors.

However, for the NoF to work well, it assumes an accurate accounting mechanism in the sense that it reflects the actual amount of resources that are consumed and provided by peers. It is important that the accounting is also autonomous, so that the NoF can continue to be deployed without requiring any special agreements between peers or access to specialized certification services.

¹ In our simulations, the perfect accounting of each task execution is equal the computational cost of the task.

² A peer performs an id-changing attack by leaving the system and immediately re-entering it using a new identification, thus (re)setting its balance to zero.

4.2. Metrics

In order to analyze the simulations, we have defined two metrics: mean response time of jobs and favor ratio. The response time is the time elapsed between the submission of a job and its completion. The relation between the mean response time and the accounting schemes is explained by the way the NoF works. Peers that provide more processing power to the grid obtain more resources when they make a request. Therefore, peers that contribute more should have smaller mean response times for the execution of their jobs.

Another important metric we have used to help us understanding the behavior of the system is the *favor ratio*

$$fr(A) = \frac{resourcesConsumed(A)}{resourcesDonated(A)},$$

where $resourcesConsumed(A)$ is the total processing power consumed by A and $resourcesDonated(A)$ is the total processing power provided by a peer A during the simulation, calculated using perfect accounting. Assuming perfect accounting and that the system has contention for resources, the NoF makes favor ratios tend to 1 for all peers in the grid, thus ensuring *fairness* [3]. However, in reality, it is not possible to implement perfect accounting. Therefore, we are interested on how "implementable" accounting schemes affect the NoF fairness. Note also that, in practice (and in our simulator), another factor that influences the favor ratio is the abortion of tasks. A peer aborts all foreign tasks when it receives a local job. When this happens, the consumer sees no value in the incomplete favor, and the provider gains no credit in its eyes. Therefore, even for perfect accounting, *favor ratios* tend to converge to a value smaller than 1 in our simulator.

4.3. Statistical Sampling

As evaluation scenarios are generated randomly, each setting evaluated must have enough samples for the results to be statistically meaningful. To do that, we chose a confidence level of 95% and estimated the confidence interval based on the results of 20 sample scenarios. Given our confidence level, in the worst case we needed to collect at least 280 samples. To keep it simple, each setting evaluated involved the execution of 280 scenarios.

As it would take too long to execute all simulations in a desktop computer, we used the OurGrid itself to access hundreds of machines in different administrative domains and run the simulations much quicker. Overall, using OurGrid enabled us to run simulations about 40 times faster than using a single machine.

4.4. Performance Evaluation

This section presents the results of the experiments we conducted in order to analyze the performance of the three accounting schemes. To do that, we have defined three settings (each corresponding to 280 scenarios simulated). In all scenarios simulated, we ensured that the load submitted to the grid was high enough to create contention in the grid. This is important because if there is a surplus of resources in the grid, then prioritizing peers has no clear effect, as everyone gets the resources they need from the system. As we are interested in observing how the NoF behaves with different accounting schemes, the system must have, on average, more than one peer interested in the resources available. To cause such contention, the inter-arrival time of jobs at peers follow a uniform distribution whose average is calculated by dividing the total amount of work required to execute an average job by the sum of the power of all the resources in an average peer. Due to the probabilistic nature of job arrival, peers are

sometimes idle and sometimes need to ask for more resources from the grid.

The first setting describes a grid consisting of peers that contain resources with different powers, to which tasks of different sizes are submitted. This setting corresponds to the *most frequent case we expect to find in practice*, in which the grid is formed by peers with heterogeneous resources, and jobs vary in size.

The second setting describes a situation in which peers have the same number of resources, those resources have the same power, but tasks vary in size. Although unlikely in practice, this represents a *disadvantageous setting for relative accounting*, because the variation in the size of tasks introduces errors in the estimation of the relative power of peers. Moreover, since resources have the same power, the time-based approach should have a performance equivalent to the perfect accounting mechanism.

The last setting shows an *accounting attack*, a situation in which half of the peers behave maliciously. They attack the accounting mechanism by pretending to have twice the number of resources that they actually have by running two tasks per resource simultaneously. This setting analyzes the robustness of the accounting schemes.

4.4.1 Setting 1: Common Case

In this setting we try to capture the heterogeneity of resources and tasks found in a grid. Thus the power of the resources and the size of the tasks vary. However, we keep the total computing power owned by each peer fixed. This means that peers are able to contribute the same amount to the grid and therefore should get the same amount of resources when they make a request. With perfect accounting, this should be translated into all peers having similar mean response times and a favor ratio that is close to 1.

We define three types of peers: those with mean resource power 4, 10, and 16. In the simulations we have a grid with 15 peers, 5 peers of each type. The power of a resource in these types is given by the uniform distributions $U(2,6)$, $U(5,15)$ and $U(8,24)$, respectively. In [14], Kee et al. show that it is reasonable to assume that the grid processors follow a uniform distribution. In addition, the sum of the power of all resources in each peer is 250. Note that this implies that peers have different number of resources.

All jobs submitted to the peers in this setting consist of 250 tasks, each task with its size varying according to a uniform distribution $U(200,600)$. (We also experimented with exponentially distributed tasks and obtained similar results.) As to saturate the peers (on average), the inter-arrival time of jobs at each peer follows the uniform distribution $U(1, 799)$, and 1,000 jobs were submitted to each of them.

Figure 1 shows the average response time for each type of peer (mean resource power 4, 10 and 16) in the simulations. The fact that peers are able to get the same amount of resources from the grid if everyone accounts correctly is reflected in the very small variation in the mean response times when using the perfect accounting scheme. The reason that there is any variation at all is that peers with slower resources are more prone to abort tasks of other peers running in their resources, as there is a higher probability that local jobs will be submitted during the execution of these tasks.

On the other hand, the time-based accounting is strongly biased towards peers with slower resources. When a resource takes longer to process a task, the favor is (unfairly) assigned a higher value. Moreover, this difference is very high: peers with slower resources have their jobs processed on average 71.6% faster than those with the fastest resources.

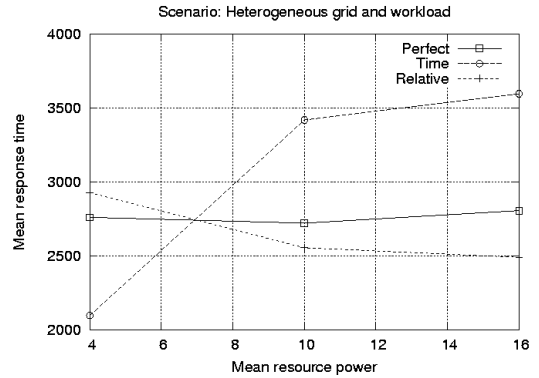


Figure 1. Power vs. mean response time for setting 1

Finally, the relative accounting scheme is slightly biased towards peers with faster resources, but it is much closer to the perfect scheme. Under this scheme peers with faster resources have mean response times 17.6% better than peers with slower ones.

Table 1 shows the favor ratios for each type of peer and each accounting scheme. These results confirm the previous analysis. The mean favor ratio for the perfect and the relative accounting schemes are very close, with the relative accounting scheme being slightly more favorable to peers with faster resources, when compared to the perfect accounting. On the other hand, for the time-based approach there is an enormous discrepancy between the mean favor ratio of the peers with mean resource power 4 and 16, with the peers with slower resources being better off than peers with faster resources.

Table 1. Favor ratio for peers in setting 1

	Mean/Std Dev of fr for peers with power 4	Mean/Std Dev of fr for peers with power 10	Mean/Std Dev of fr for peers with power 16
Perfect	0.908 / 0.114	0.981 / 0.099	0.994 / 0.101
Relative	0.872 / 0.122	0.995 / 0.079	1.020 / 0.075
Time	1.051 / 0.084	0.866 / 0.095	0.830 / 0.100

This setting gives evidence that the relative accounting scheme is suitable for a peer-to-peer grid, as it has behavior close to that of a perfect accounting scheme. Moreover, the small bias towards peers with faster resources has the side-effect of providing an incentive for peers to make the fastest possible resources available. Since the bias is not large, we believe it should not be enough to put off peers that cannot afford to own fast resources.

4.4.2 Setting 2: Unfavorable Case

In this setting, we have 15 peers of the same type. Each peer has 25 computational resources, and all resources have the same power, which is 10. However, the jobs submitted to each peer contain tasks that are very heterogeneous in size, varying according to an exponential distribution $f(x) = 0,0025e^{-0,0025*x}$, which has mean equal to 400. Each peer received 1,000 jobs consisting of 250 of those tasks. We created contention in the grid by submitting those jobs with inter-arrival times following a uniform distribution $U(1, 799)$.

This setting is a special case in which accounting a favor correctly is equivalent to accounting the time the task took to be processed in a resource. In such a case, we expect to see the best

performance of the time-based scheme. Moreover, because of the large variation in the size of tasks, this scenario is particularly unfavorable to the relative accounting scheme.

Table 2 shows the response times and favor ratios for each type of accounting scheme. Since all peers in this scenario have the same power and receive the same mean workload, we are here more concerned about the standard deviation of these metrics, in detriment of their means, which should be very close.

As expected, the perfect and the time-based accounting schemes have very similar performance. Considering the response time metric, its standard deviation for the relative accounting scheme is only 3.44% higher than those for the other approaches. When considering the favor ratio metric, the differences are bigger. The standard deviation of the favor ratio (which will be large if some peers have favor ratios far from 1) is 40% higher for the relative accounting scheme than those for the perfect and time-based schemes.

Table 2. Response time and favor ratios for peers in setting 2

	Mean/Std Dev of Response Time	Mean/Std Dev of Favor Ratio
Perfect	2,921.6 / 2,783.6	0.9524 / 0.125
Relative	2,951.2 / 2,879.4	0.9193 / 0.175
Time	2,922.1 / 2,782.6	0.9523 / 0.125

We note, however, that most applications we are aware of, are not formed by tasks whose sizes can be modeled by an exponential distribution. They tend to have tasks whose sizes are more uniformly distributed, instead. Our experiments for this scenario considering tasks' size following a uniform distribution $U(100, 700)$ show similar results for the response time metric, whereas the standard deviation of the favor ratio is only 14% higher for the relative accounting scheme when compared to those of the other schemes.

4.4.3 Setting 3: Accounting Attack

This setting gauges the behavior of the accounting schemes when peers try to take advantage of the system by providing resources of poor quality. It is important that consumer peers are able to identify these resources. If providers can receive the same credit, no matter whether the resource is fast or slow, they have an incentive to provide the slowest possible ones, and the system collapses, a phenomenon known as the market for lemons [1].

The setting is a grid with 20 peers, each with 25 machines whose power varies following the uniform distribution $U(4,16)$. However, 10 of such peers run 2 tasks on each machine, appearing for the grid to have 50 machines of power $U(2,8)$ ³. We name these peers slower, whereas the other 10 peers, which run 1 task per machine, are called faster. Jobs arrive every $U(1,799)$ time units, each carrying 250 tasks, each task demanding $U(200,600)$ time units to execute.

As Figure 2 shows, when the perfect accounting is used, the slower peers get no benefit from their behavior. In fact, as they take longer to process tasks, they are more prone to abort tasks from other peers. This makes them get less credit in the grid, which is reflected in their slightly worse mean response time when compared to the faster peers.

³ In practice this problem is exacerbated by the fact that the strategy followed by the malicious peers may introduce trashing in a resource when the tasks that share the resource cannot all fit in main memory, thus decreasing the power of the grid as a whole.

The relative accounting scheme has similar behavior. However, the difference between the mean response time of the faster and slower peers is a little larger. The mean response time for faster peers is 7.13% shorter than when using the perfect accounting scheme. The mean response time for the slower peers, on the other hand, is 2.63% longer.

For the time-based accounting, we see that slower peers get an advantage when compared to the faster peers, creating an incentive for peers to adopt this strategy. The time-based approach gives two advantages to the slower peers. First, it enables the malicious peer to run twice the number of tasks non-malicious peers can run simultaneously and thus to provide more favors to other peers. Second, it increases the execution time of each task, which benefits the malicious peer, because other peers value its favors based on how long it takes to run their tasks.

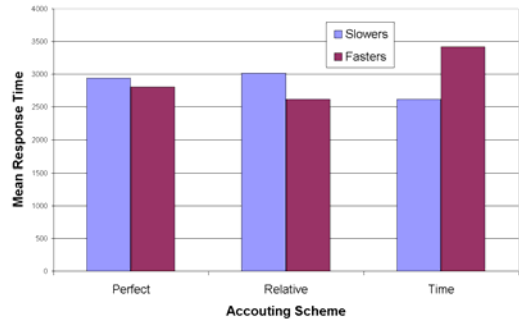


Figure 2. Mean response time for setting 3

Looking at the favor ratios in Table 3 we see how the time-based accounting makes the grid prioritize the slower peers. They get more for the resources they provide to the grid. Again, the relative accounting scheme gets very close to the perfect one, with a standard deviation of favor ratio for all peers only 0.0077 larger than that for the perfect scheme.

Table 3. Mean favor ratio for peers in setting 3

	Mean/Std Dev of f_r for slower peers	Mean/Std Dev of f_r for faster peers	Mean/Std Dev of f_r for all peers
Perfect	0.931 / 0.122	0.976 / 0.111	0.953 / 0.119
Relative	0.911 / 0.132	0.989 / 0.107	0.950 / 0.126
Time	1.003 / 0.117	0.833 / 0.117	0.918 / 0.145

This setting shows that using the relative accounting scheme makes the peer-to-peer grid robust against the problem of the market for lemons. It provides consumers with accurate enough information to prevent provider peers serving low-quality resources from gaining an advantage over those providing high-quality ones. Although the relative accounting scheme introduces a larger difference in the favor ratio mean between faster and slower peers than a perfect scheme, this difference (8.5%) is not significantly high. Since it is infeasible to deploy a perfect accounting scheme in a peer-to-peer grid, we believe that the relative accounting scheme is a good enough approximation.

5. RELATED WORK

The problem of accounting on grids has been considered before. The major difference between our solution and previous works is that we specifically target accounting among parties that

do not trust each other, whereas other proposals assume that parties exchange correct measured data.

In fact, in order to measure the resource utilization and power of resources, GridBank [5] and the DataGrid Accounting System (DGAS) [12] use resource meter agents deployed throughout the grid. We argue that deploying the functionality necessary for collecting utilization data faces serious trust issues for peer-to-peer grids. SweGrid Accounting System (SGAS) [10] uses only the time that the resources are available to run applications as accounting metric. In SGAS case, the accounting is simplified because all resources are equal. However, as we showed in our results, for a more heterogeneous grid, using only time benefits providers of slow resources. In short, in GridBank, SGAS and DGAS the accounting is done in the provider side and sent to the consumer or a third-party bank. In a peer-to-peer grid, free-riding providers can send fraudulent accounting information, misleading the consumer. To circumvent this, we autonomously account the resource allocations, so there is no need of exchanging information among parties.

Also, Thigpen et al. [18], DGAS and GridBank assume that there is a previous negotiation between the consumer and the provider before the resource utilization. To help the negotiation, they mention the need to estimate the cost for running a job prior to the use of the resource. This estimation could be supplied by the user or estimated using historical information. Instead, in our scheme, consumers and providers do not negotiate before execution and autonomously estimate the relative peer power during the job execution, what greatly simplifies the use of an accounting scheme from the user's point of view.

6. CONCLUSIONS

We have analyzed three accounting schemes to support the implementation of an autonomous reputation-based allocation mechanism for peer-to-peer grids. The perfect scheme requires timely and accurate information on the power that all resources in the grid are able to deliver at any given time. It is very unlikely that such a scheme can be implemented in practical peer-to-peer grid systems. The time-based scheme, although very simple to implement and deploy, always gives a noticeable benefit for peers with slower resources in detriment to those with faster resources. As a result, this accounting scheme is vulnerable to a simple attack in which malicious peers voluntarily reduce the power of their resources by executing more than one task at a time in each resource.

The relative accounting scheme behaves closer to the perfect accounting scheme and is also very simple to implement and deploy. Even for the less favorable (unlikely in practice) setting in which the computing resources that form the grid are homogeneous and the workload submitted to the grid is heterogeneous (see Section 4.4), our experiments have shown that the relative accounting scheme is only slightly worse than the perfect accounting. The main difference when comparing our accounting scheme against the perfect accounting scheme is that it is slightly biased towards peers with faster resources. We believe, however, that this will not have a negative impact on the grid. On the contrary, it makes in the best interest of the peers to provide their most powerful resources to the grid. Peers therefore have an incentive to increase their local power and as a consequence the power of the grid as a whole.

7. ACKNOWLEDGEMENTS

This work was partially developed in collaboration with HP Brazil R&D. Francisco Brasileiro and Walfredo Cirne would like to thank the financial support from CNPq/Brazil (grants

300.646/96 and 302.317/03, respectively). Thanks also to Katia Saikoski for the comments and suggestions.

8. REFERENCES

- [1] E. Adar and B. Huberman. *Free riding in Gnutella*. First Monday 5 (10), October 2000.
- [2] N. Andrade, M. Mowbray, W. Cirne and F. Brasileiro. *When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System?* 4th GP2PC, USA, April 2004.
- [3] N. Andrade, F. Brasileiro, W. Cirne and M. Mowbray. *Discouraging Free-riding on a Peer-to-Peer Grid*. Proceedings of HPDC13, June 2004.
- [4] G. Akerlof. *The market for lemons: quality uncertainty and the market mechanism*. Quarterly Journal of Economics 84 (3), 488-500, 1970.
- [5] A. Barmouta and R. Buyya, *GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration*. 26th ACSC2003, Australia, February 2003.
- [6] A. Butt, R. Zhang, Y. Hu, *A Self-Organizing Flock of Condors*. Supercomputing'2003.
- [7] F. Cappello, S. Djilalia, G. Fedak, T. Herault, F. Magniettea, V. Nérib and O. Lodygensky. *Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid*. Future Generation Computer Systems, 21(3):417-437, March 2005.
- [8] W. Cirne, F. Brasileiro, N. Andrade, R. Santos, A. Andrade, R. Novaes, M. Mowbray. *Labs of the World, Unite!!!* UFCG TR 01/05. <http://walfredo.dsc.ufcg.edu.br/resume.html>
- [9] B. Cohem. *Incentives Build Robustness in BitTorrent*. P2P Econ'03, June 2003.
- [10] E. Elmroth, P. Gardfjell, O. Mulmo, and T. Sandholm. *An OGSA-Based Bank Service for Grid Accounting Systems*. In J. Wasniewski et. al. (eds). Applied Parallel Computing. State-of-the-art in Scientific Computing. Springer Verlag, Lecture Notes in Computer Science, 2004.
- [11] I. Foster and A. Iamnitchi. *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*. 2nd IPTPS'03, February 2003, Berkeley, CA.
- [12] A. Guarise, R. Piro, A. Werbroeck. *DataGrid Accounting System - Architecture v 1.0*. April 14th 2003. Technical Report. <http://eu-datagrid.web.cern.ch/eu-datagrid/>.
- [13] D. Hughes, G. Coulson and J. Walkerdine, *Free Riding on Gnutella Revisited: The Bell Tolls?* IEEE Distributed Systems Online, Volume 6, Issue 6, June 2005.
- [14] Y. S. Kee, H. Casanova, A. Chien. *Realistic Modeling and Synthesis of Resources for Computational Grids*. In *Proceedings of SC'04 Pittsburgh, PA*, November 2004.
- [15] V. Lo, D. Zhou, D. Zappala, Y. Liu, and S. Zhao. *Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet*, IPTPS 2004.
- [16] S. Saroiu, P. Gummadi and S. Gribble. *A Measurement Study of Peer-to-Peer File Sharing Systems*. MMCN'02, USA, January 2002.
- [17] K. Shudo, Y. Tanaka and S. Sekiguchi. *P3: P2P-based Middleware Enabling Transfer and Aggregation of Computational Resources*. 5th GP2PC, May 2005.
- [18] W. Thigpen, T. Hacker, B. Athey and L. McGinnis. *Distributed Accounting on the Grid*. Proc. 6th JCIS, 2002.