Scalable Computing: Practice and Experience
Volume 11, Number 4, pp. 385–392. http://www.scpe.org
ISSN 1895-1767
© 2010 SCPE

REPUTATION-BASED MAJORITY VOTING FOR MALICIOUS GRID RESOURCES TOLERANCE

AHMED BENDAHMANE, MOHAMMAD ESSAAIDI, AHMED EL MOUSSAOUI AND ALI YOUNES*

Abstract. Recently, distributed computing infrastructures such as grid computing, have witnessed huge developments which have a very important impact on computationally intensive scientific, industrial and engineering problems. Collaborative computing, resources access and sharing facilitated by grid computing systems amplifies concerns about cyber attacks and misbehaviors of grid resources (resources provider). In many cases, an organization may send out jobs for remote execution on untrustworthy machines. Since resource sharing and cooperation among different administrative domains are among the central goals of grid computing, it is necessary to guarantee that, among the shared resources, there are no malicious resources interested in invalidating or corrupting the job results. In order to guarantee reliable jobs execution in grid systems, we have developed a new approach for malicious grid resources tolerance using reputation to improve the efficiency of majority voting mechanisms. The grid broker service investigates the trustworthiness of the grid resources used in the voting procedure; this investigation is based on distributed and hierarchical reputation management system which maintains a reputation value for each grid resource based on its historical behavior. The validation of the result is then decided based on the grid resources reputation level.

Key words: grid computing, malicious resources, tolerance, majority voting, reputation

1. Introduction.

A grid system is a platform that provides access to various computing resources owned by different institutions through the creation of virtual organizations [1]. A grid virtual organization (VO) allows a seamless aggregation of computational resources in multiple administrative domains into a single pool. The users can lease for a certain time-frame bundles of software, CPU cycles, bandwidth and storage space in order to execute computationally intensive tasks, according to a service level agreement (SLA) between the user and the grid broker service. This broker is acting as a service intermediate between users and grid resources. This brokering role can either happened in strong structured grids, with long-lasting VOs inside the VO boundaries or in volatile grid environments, with very dynamic VOs, where the grid resources can dynamically join and leave in the middle of an execution without impediment. In the second scenario, which is very close to the grid design adopted by the XtreemOS project [15], the execution of a task may be halted arbitrarily. It is then mandatory for VO manager to tolerate these misbehaviors in order to assure the security of the user jobs execution.

Moreover, as the grid resources are connected through the Internet, and with the fusion of web services and grid technologies, and due to the inherently dynamic and complex nature of grid system, a range of new security threats and vulnerabilities are coming to the fore with the broadening scope of grid computing [3], and especially the grid resources has become a subject of several unknown attacks. Introduction of grids in the commercial sector has open up new ventures in the business arena [8]. However, the contemporary grid application domains require many protections from malicious entities. These applications and their underlying infrastructure have to protect the critical information about their owners; provide safeguards to the economic interests of the enterprises using the grid system; and win the confidence of the society which is already doubtful of the digital data processing [5].

Conventional grid security mechanisms [21] can ensure several security services. Some of them guarantee communication confidentiality between grid nodes [17, 18] while others investigate resources access [19, 20]. As far as data integrity is concerned, most mechanisms that have been proposed to guarantee it will do it only during the transmission; employing authentication and encryption technologies (as in Globus system) [22, 23, 24]. However in order to really guarantee data integrity in a grid system, it is also needed to ensure the integrity of the job result processed by the grid resources. Hence, it is necessary to protect the applications in process in the grid against possible malicious behavior of attacked resources that can affect the efficiency of the grid system by returning erroneous jobs results.

In addition, the existing attacks/intrusions detection and prevention mechanisms are not efficient against unknown attacks. Therefore, another kind of defense mechanism is needed to implement the precaution against all possible attacks/intrusions. This creates the requirement that grid environments should detects and tolerates the grid resources attacked by an adversary, which tamper with the computation and return corrupted results, in order to guarantee reliable execution in such a suspicious environment.

*Information and Telecommunication Systems Laboratory, Faculty of Science, Tetuan, Morocco, ({a.dahman, elmoussaoui, younes.ali}@uae.ma, essaaidi@ieee.org).

In this paper, we propose a new approach for tolerating malicious grid resources by using majority voting mechanisms, and then investigate the trustworthy of the grid resources which are used in voting procedure. This investigation is based on reputation management system. And the validation of result is decided based on the reputation level of the grid resources.

The remainder of this paper is organized as follow. Section 2 presents our motivation of using sabotage tolerance techniques in grid. Section 3 survey the background of these techniques. In section 4, we present our vision of basic components of grid system and the possibility of attacking these components. Section 5 describes our approach of using majority voting with reputation. Section 6 presents how the grid broker service will build the reputation of grid resources. At last, section 7 concludes the paper.

2. Motivation. Sabotage tolerance is gaining importance in grid environments notably in the situation where different grid domains have conflicting interests. The term sabotage tolerance was originally coined in the specific area of Desktop Grids, where voluntarily Internet users contribute to the grid with computing cycles. Because everyone can take part in such a grid, the environment started loosing its trustworthiness. Since computations run in an open and un-trustable environment, it is necessary to protect the integrity of data and validate the computation results. Sabotage tolerance techniques need to be employed mainly for the detection of malicious entities who may submit erroneous results.

In a classical grid, sabotage tolerance is not an issue because the grid environment is trustable, in the sense that someone (grid node administrator) controls strictly the grid resources and their ownership. When a grid resource is malfunctioning, the grid owner is notified. However, if the grid scales at the size of the Internet or spans over multiple administrative domains without a hierarchical subordination of control and ownership, an attacker might corrupt the grid resources by exploiting some of its vulnerabilities, then it is likely that grid resources report erroneous job results. Sabotage tolerance becomes mandatory, for the protection of both the grid users and other grid contributors.

Sabotage tolerance techniques are applied in grid systems that employ the master-worker computational model [13]. This grid model is not restrictive and maps well on the wide variety of grids. This model can be summarized as a server (referred further as the master) that distributes work units of an application to grid nodes (workers). A result error is any result returned by a worker that is not the correct value or within the correct range of values [10]. Sabotage-tolerance techniques imply detecting result errors, which are very important as they can undermine long computations that have been executing during weeks or even months by several workers [4]. The error rate ε is defined as the ratio of bad results or errors among the final results accepted at the end of the computation. Thus, for a batch of N work units with error rate ε, the master expects to receive εN errors. For every application, the master employs some sabotage-tolerance mechanism for obtaining an acceptable error rate ε_{acc} with regard to its application. If a grid user comes with an application divided on a big number of tasks (e.g. 10 batches of 100 work units each) and it requires a global error rate of 10^{-2}, the sabotage tolerance technique should provide with a work unit error rate of about 10^{-5} [10]. Many applications, especially the ones from computational biology and physics require bounds on the error rates, as the correctness of the computed results is essential for making accurate scientific conclusions.

3. Background and related work. In this section, we provide a short overview for the principal sabotage tolerance techniques in Desktop Grids. Sabotage tolerance techniques for Desktop Grids can be classified in four big classes [4]: reputation systems, replication with voting, sampling, and checkpoint-based verification.

Reputation systems [26] are commonly used to estimate the reliability of grid resources based on past interactions. The concept of trust-aware resource management for the Grid was proposed in [27], where a technique is presented for computing trust ratings in a Grid using a weighted combination of past experience and reputation. Zhao and Lo [16] propose augmenting peer-to-peer cycle sharing systems with a reputation system to reduce the degree of replication required to verify results. GridEigenTrust [28] combines this trust-computation technique with the EigenTrust reputation system [29] to provide a mechanism for rating resources in a Grid. These existing reputation systems have focused on correctness as the primary metric, and have dealt mainly with binary trust values. The key concept in our reputation model is the combination of the credibility, availability, and security level to compute the overall grid resources reputation.

Replication with majority voting [13] is widely used in the BOINC Desktop Grid platform [2]. The master distributes $2m-1$ replicas of a work unit to workers and when it collects m similar results, it accepts that result as being the correct one. Each collected result is seen as a vote in a voting pool with $2m-1$ voters and with majority agreement being the decision criteria. The error rate of this method is determined by the number of identical results required (m), which is a measure of redundancy. High levels of redundancy provide very low error rates (less than 10^{-5}). The main benefit of the method is its simplicity, while the big drawback is the fact that it wastes a lot of resources.

Sampling-based techniques are developed to overcome limitations of replication, especially redundancy. Within sampling, the master determines the trustworthiness of the workers by verifying them only on a few samples of their results. The basic sampling is the naïve one [14], where the master sends probes with verifiable results to workers. If workers respond well to the probes, they are considered trustworthy. The main drawback is the workers can easily recognize the probes and respond well to them, while cheating on the ordinary work units. The probing by spot checking is introduced by Sarmenta [13]. A credibility based mechanism is presented to protect a volunteer computing environment against malicious users. Probes, named now spotters, are tasks with known results. If a worker fails to compute correctly a spotter, it will get blacklisted and all its results will be invalidated. Based on spot-checking, Sarmenta defines the credibility of a worker and a result. The credibility of a worker is an estimate of the probability the worker will return a correct result. The credibility of a result is the conditional probability that the result originating from a worker will be accepted as correct. Sarmenta approach is essentially a centralized one, the fact that adds an overhead to the central resource responsible for the tests (every spotter job is also processed by the central resource). In our approach, which is based on a distributed and hierarchical paradigm, we delegate the spot-checking operations to other resources known to be trustworthy; we assume that there are some highly pre-trusted resources in the grid system we call reputation agent. Moreover, we will use these highly pre-trusted resources to store the credibility value, and then to reduce the traffic and the processing effort by a centralized resource according to the tests results.

Quizzes [16] are an improvement to basic probing. With this method, the master sends to workers batches with work units and it places the probes inside of those batches. Given the actual required error rate, the master can compute the number of quizzes to place inside a batch. A drawback of probing is the fact the master should possess some heuristic in order to generate the probes and make them to resemble with the real work units. Because this is a difficult task, the master can use actual work units as probes [13]. The master verifies (using replication) only a sample of results and if a worker is caught cheating all its previous results are invalidated.

Checkpoint based verification addresses the problems with sequential computations that can be broken in multiple temporal segments (S_{t1}, S_{t2}, . . . , S_{tn}). At the end of each segment a checkpoint is submitted to a stable storage. The checkpoints are stored locally to allow recovering the long tasks from faults. After finishing a task, the worker sends back the result along with a list of hashes for the checkpoints. In the basic checkpoint verification [11], the master randomly selects a checkpoint time S_{ti} for a task and asks the worker to deliver its local checkpoint $C(S_{ti})$. Then the master computes the task from S_{ti} up to the next checkpoint and compares the results with the hash value submitted by the worker for $C(S_{ti+1})$. If the hash verification succeeds, then the worker passed the verification. In the distributed version of the checkpoint verification, the master selects a third worker for verification purposes and let the verifier to compute the checkpoint verification. The distributed version has the advantage of not overloading the master with a lot of verification tasks. The error rate of this method strongly depends on the number of verified checkpoints - i. e. a high percentage of verified checkpoints yield a low error rate, for the cost of increased computation (redundancy) and bandwidth.

4. Model and assumption.

Fig. 4.1 shows the basic components of a grid system of interest, which consists of N Virtual Organizations (VOs), where each VO contain a set of resources, services and middleware, applications, and users, connected by a local network and the Internet (Fig. 4.2). From the service point of view, these components are considered as resources provider and services provider. The former owns computational nodes or other physical resources, and has root privilege. It decides whether and when resources should contribute and the quantity of available resources that are needed for job execution. The number of resources is dynamic and undetermined. Besides, each resource provider may require different authentication, authorization mechanisms and security policies. The second is the owner of the software solutions and the information databases that are deployed on a resource provider's assets. The software solution is an individual service or a combination of services to solve user's submitted jobs or requests. In combination situation, the service provider then comes to be a user that requests other services.

In grid computing environment, a client is a parallel job submitter who requests results, he/she can submit a job to grid broker service (after SLA establishment), which determines where and when to run the job. The broker primarily mediates user access to grid resources by discovering suitable services and resources provided by VOs and deploying and monitoring job execution. In the other hand, each VO in the grid has a single VO

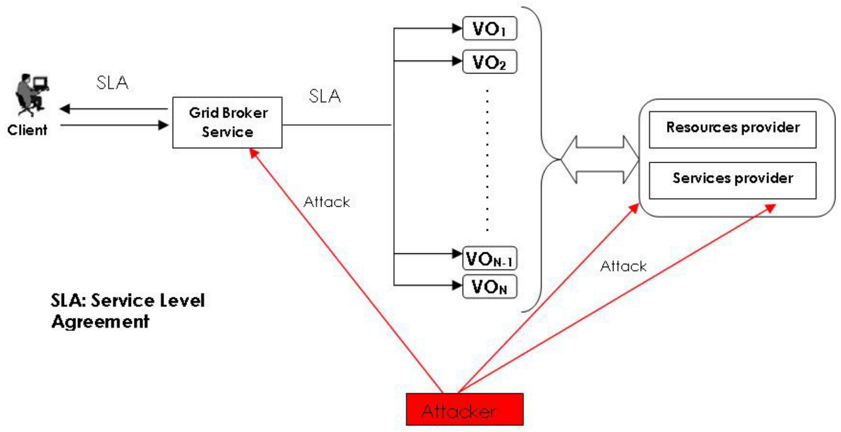

FIG. 4.1. *Basic components of a grid system*

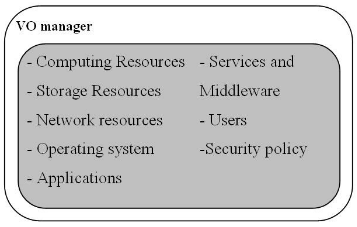

FIG. 4.2. *Virtual organization (VO) components*

manager, which searches for available services. When it finds an interesting service, it negotiates with the service provider to allow access to the service from their VO. Once the service provider authorizes the use of the service, the service becomes available at this VO. The job is then divided into subjobs that have their own specific input data. The subjob is called a task. The broker distributes tasks to selected grid resources (computing resources) using scheduling mechanisms. When each grid resource subsequently finishes its task, it returns the result of the task to the broker. Finally, the broker returns the final result of the job back to the client.

In our study, the VO manager includes a reputation agent component, responsible for spot-checking operations, and for the reputation value computation for each grid resource within the VO. The reputation values are then returned to the broker.

In this execution scenario of jobs, there are several possible attacks to grid resources which might tamper with the computation and return corrupted results (grid resources takeover by an unauthorized user can result to malicious acts like corruption of job data, job termination etc.). Such attacks are usually related to security holes of the services and resources provider [9]. Denial of Service (DoS) attacks [7] may also disrupt job completion or temporarily prevent access to job output by cutting a site off the Resource Broker that has delegated the job and is waiting for the output.

5. Malicious grid resources tolerance using reputation. Attacks on grid computing, as on any other system, rely on a large number of different vulnerabilities in grid security. Therefore, these attacks become an increasingly prevalent form of security threats, and the problem appears to be quite difficult to solve, especially for attacks which can corrupt the grid resources and make them provide erroneous results of execution. Nevertheless, there exist several defense mechanisms that have been proposed to detect and prevent these attacks, these mechanisms can stop known attacks, but can not defend against new attacks which can happen in grid system. To this end, tolerance mechanisms for malicious grid resources become necessary to provide a safe result of job execution. As a solution, we propose an approach of tolerating these types of attacks by using sabotage tolerance techniques.

The proposed method is based on reputation to improve the efficiency of majority voting mechanisms. With majority voting the grid broker service (i. e. master) decide about the trustworthiness of a result immediately after having the result of all replicas of a task (i. e. work unit). This approach tolerates a certain number of incorrect results in a vote. However, it does not resist when, for example an adversary gets control through a virus over a large number of computing resources (i. e. workers), and make them return the same result. In our approach, the broker will postpone the decision until the moment it gathers enough information to infer the trustworthiness of the result. To achieve this information, we will use existing reputation lists of different computing resources in VO.

5.1. Problem formulation. Let us assume that each task is replicated n times and allocated to several computing resources C_i, so that a broker can collect m different results V_j, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. Each computing resource has its reputation value R_i, this reputation is collected by a grid broker service, which contains the reputation lists for all computing resources C_i. The reputation is a value in the range between 0 and 1. In Sect. 6 we will show how the reputation agent will compute the reputation value.

To make a decision about which result V_j is trustworthy (i. e. accepted result), the grid broker service utilizes a majority voting based on reputation decision criteria.

Let the relativity vector $Re(V_j)$ represents The aggregated vote of all computing resources that return the same result.

$$Re(V_j) = [T(V_j, C_1), T(V_j, C_2), \ldots, T(V_j, C_n)] \tag{5.1}$$

Where $T(V_j, C_i)$ is the relationship between the result V_j and the computing resource C_i. It is calculated as follow.

$$T(V_j, C_i) = \begin{cases} 1, & \text{if } C_i \text{ compute } V_j \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

We define the result reputation $RR(V_j)$ of a given result V_j as the summation of reputations of the computing resources returning the result V_j.
For each result V_j:

$$RR(V_j) = \sum_{i=1}^{n} T(V_j, C_i) R_i \tag{5.3}$$

To make a decision we fixe a positif threshold value $\lambda < 1$ and we pick the maximum of $RR(V_j)$; $(j = 1, \ldots, m)$.

$$\begin{cases} if \quad max(RR(V_j)) > \lambda \sum_{i=1}^{n} R_i; & \text{the result corresponding to this maximum is accepted} \\ Otherwise; & \text{the grid broker service should decide for further replication} \end{cases}$$

To avoid the possibility that a set of computing resources with a low reputation could undermine the result, we impose the following condition.

$$R_i = \begin{cases} 0 & \text{, if } R_i < \Theta \\ R_i & \text{, otherwise} \end{cases} \tag{5.4}$$

Where Θ represents the minimum reputation value a computing resource should have for its results to be taken in consideration.

6. Computing reputation. In this section, we present how the grid broker service will build the reputation value of each computing resource (or grid resource).

The reputation of an entity is an expectation of its behavior based on other entities observations or information about the entities past behavior at a given time [6]. The reputation management services are responsible for evaluating the reputation of grid resources, services, and users inside a VO. In our vision, the grid broker service builds the reputation of each computing resource through its credibility, availability, and security level.

6.1. Credibility. The credibility represents the likelihood of a particular object of the system to be operating properly [12]. The reputation agent the credibility of the computing resource C_i by passing spot checking (see Sect. 3). If f is the proportion of compromised computing resources, the credibility of a computer resource which correctly computed k_i spotters (tasks), when blacklisting is not used, will be computed by [13]:

$$CR(C_i, k_i) = \begin{cases} 1 - \frac{f}{k_i}, & \text{if } K_i \neq 0 \\ 1 - f, & \text{otherwise} \end{cases} \tag{6.1}$$

With blacklisting policy [25], the malicious resources which return an erroneous result for the spotter job are immediately blacklisted, i. e. the malicious resources are never allowed to return results or to get any more jobs. In our study, we prefer to compute credibility without the blacklisting policy, because in some way it is possible that a malicious resource may change its behavior with the time. For instance, resources may be infected by a virus and after being repaired it will become reliable again. Thus, these repaired resources can participate in a computation and return a correct result, and can then improve the performance of the grid system.

After the computing resources pass enough tasks k_{min}, they succeed to obtain enough credibility (minimum credibility) to guarantee that their results are correct given the error rate ε_{acc}. First, we initialize the credibility value $CR_{init}(C_i)$ of all computing resources. The initial value is no more than the minimum credibility of the computing resource itself.

$$CR_{init}(C_i) = CR(C_i, k_{min}) \tag{6.2}$$

The credibility value is incremented in such a way that if a computing resource returns a validated result by the reputation-based majority voting described above. We propose to consider as a passed spotter each task successfully validated by the broker, using the reputation-based majority voting. Then, we update the credibility value using this equation:

$$CR(C_i, k_i) = CR(C_i, k_i + 1) \tag{6.3}$$

In the same manner, we do decrease the credibility of those computing resources whose result was not validates by the reputation-based majority voting. To this end, we update the credibility value using this equation:

$$CR(C_i, k_i) = CR(C_i, k_i - 1) \tag{6.4}$$

6.2. Availability. The Availability Ai is the ratio of the number of successful contacts to the computing resource C_i and the total number of requests. A non-available resource means that it is not accessible [30]. Thus, the computation of availability is given by:

$$A_i = \frac{N_S}{N_R} \tag{6.5}$$

Where N_S is the number of successful contacts to the computing resource, and N_R is the Number of the total number of requests.

When a computing resource joins the grid system, it may participate in the computation of several jobs. Every time, its availability value will be updated.

6.3. Security level. In our approach, we consider the security level of a computing resource as a self-protection capability.

As defined in [31], the self-protection capability of a computing resource is calculated by aggregating the security factors. The values of these factors differ in the range between 0 and 1. Based on their contributions to security, a proportion is given to each security factor as a final point aggregated to compute the self-protection capability. The security factors and their related proportions are listed in Table 6.1 [32].

Thus, the security level (SL_i) is calculated using the following equation:

$$SL_i = \frac{\sum_{s=1}^{n} P(s)L(s)}{n} \tag{6.6}$$

Where n is the total number of factors. $P(s)$ is the proportion and $L(s)$ is the value of the factor determined by the reputation agent trust strategy.

TABLE 6.1
Proportion of security factors.

Security Factors	Proportion (P)
IDS Capabilities	0.825
Anti-virus Capabilities	0.85
Firewall Capabilities	0.9
Authentication Mechanism	0.8
Secured File Storage Capabilities	0.7
Interoperability	0.6
Secured Job Execution	0.75

6.4. grid Resources reputation. The reputation R_i value of a computing resource C_i, that will be used by the reputation-based majority voting, computed by the reputation agent, is the product of the following three metrics: credibility, availability, and security level, and it is computed using the following equation:

$$R_i = CR(C_i, k_i) \times A_i \times SL_i \tag{6.7}$$

As the SL_i of a computing resource is usually stable for a period of time, when a new computing resource joins the grid, it will be calculated as the initial reputation value of the computing resource. Each result produced by the computing resource will enter the reputation-based majority voting competition. If the result is accepted, the reputation of the computing resource will be increased. If not, it will be decreased.

7. Conclusion. In this work, the proposed approach for tolerating attacks in grid resources is discussed. This approach is based on majority voting mechanisms and reputation management system. The voting result that is generated by the majority voting can be combined with the reputation of grid resources owned by VOs, and then it may be possible to decide on the authenticity of suspicious grid resources. Using reputation information with majority voting, our grid computing system can provide job results correctness. Thus, our tolerance scheme for detecting suspicious grid resources is more accurate and more reliable. And help to improve the security of grid computing system.

REFERENCES

[1] F. Berman, G. C. Fox, and A. J. G. Hey, *Grid Computing: Making the Global Infrastructure a Reality*, Wiley, Chichester, 2003.

[2] P. A. B. David, *Boinc: A system for public-resource computing and storage*, In GRID2004. The Fifth IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, Washington, 2004, pp. 4–10.

[3] Y. Demchenko, L. Gommans, C. Laat, and B. Oudenaarde, *Web services and grid security vulnerabilities and threats analysis and model*, In GRID2005. The 6th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, Washington, 2005, pp. 262–267.

[4] P. Domingues, B. Sousa, and L. M. Silva, *Sabotage-tolerance and trust management in Desktop Grid computing*, Future Generation Computer System 23(7), 2007, pp. 904–912.

[5] J. Ermisch, and D. Gambetta, *People's trust: The design of a survey-based experiment*, IZA Discussion Papers 2216, Institute for the Study of Labor, 2006.

[6] A. Farag, and M. Muthucumaru, *Towards Trust-Aware Resource Management in Grid Computing Systems*, In CC-GRID2002. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, 2002, p. 452.

[7] P. Galli, *DoS attack brings down Sun Grid demo*,(2006, Mars). Available: `http://www.eweek.com/article2/0,1895,1941574,00.asp`

[8] ———, *Commercial grid solutions, Grid Computing Planet* , (2006, December, 18). Available: `http://www.gridcomputingplanet.com/resources/article.php/933781`

[9] N. Jiancheng, L. Zhishu, G. Zhonghe, and S. Jirong, *Threat analysis and Prevention for grid and web security services*, InProc. of Eighth ACIS. International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, pp. 526–531.

[10] D. Kondo, F. Araujo, P. Malecot, P. Domingues, L. M. Silva, G. Fedak, , and F. Cappello, *Characterizing result errors in Internet Desktop Grids*, In Euro-Par2007. LNCS, vol. 4641, Springer, Heidelberg, 2007 pp. 361–371.

[11] F. Monrose, P. Wyckoff, and A. D. Rubin, *Distributed execution with remote audit*, The Network and Distributed System Security Symposium, NDSS. The Internet Society, San Diego, 1999.

[12] A. C. Oliveira, L. Sampaio, S. F. Fernandes, and F. Brasileiro, *Adaptive Sabotage-Tolerant Scheduling for Peer-to-Peer Grids*, In LADC2009. Fourth Latin-American Symposium on Dependable Computing, IEEE Computer Society, Joao Pessoa, 2009, pp. 25–32.

[13] L. F. G Sarmenta, *Sabotage-tolerance mechanisms for volunteer computing systems*, Future Generation Computer Systems 18(4), 2002, pp. 561–572

[14] D. Wenliang, J. Jing, M. Mangal, and M. Murugesan, *Uncheatable grid computing*, In ICDCS2004. The 24th International Conference on Distributed Computing Systems, IEEE Computer Society, Washington, 2004, pp. 4–11.

[15] E. Y. Yang, B. Matthews, A. Lakhani, and Y. Jégou, *Virtual organization management in XtreemOS: an overview*, Towards Next Generation Grids, Proc. of the CoreGRID Symposium, Rennes, France, Springer, Heidelberg , 2007.

[16] S. Zhao, V. Lo, and C. GauthierDickey, *Result verification and trust-based scheduling in peer-to-peer grids*, In P2P2005. The 5th IEEE International Conference on Peer-to-Peer Computing, IEEE Computer Society, Washington, 2005, pp. 31–38.

[17] A. Setiawan, D. Adiutama, J. Liman, A. Luther, and R. Buyya, *Gridcrypt: High performance symmetric key cryptography using enterprise grids*, In PDCAT 2004. Parallel and Distributed Computing - Applications and Technologies, 2004, pp. 872–877.

[18] S. Song, K. Hwang, and J. Lv, *Grid security enforcement with trust integration over minimal vpn tunnels*, Technical Report TR 2004-10, USC Internet and Grid Computing Lab, 2004.

[19] J. Silva and L. Gaspary, *Uma arquitetura de controle de acesso baseado em politicas para grades computacionais peer-to-peer*, In Workshop of Peer-to-Peer in the 23rd Brazilian Symposium on Computer Networks, 2005, pp. 37–48

[20] A. Goldchleger et al, *Integrade object-oriented grid middleware leveraging idle computing power of desktop machines*, In Concurrency and Computation: Practice and Experience, 2004, pp. 449–454.

[21] A. Bendahmane, M. Essaaidi, A. El Moussaoui, and A. Younes, *Grid computing security mechanisms: State-of-the-art*, International Conference on Multimedia Computing and Systems, Ouarzazate, 2009, pp. 535—540.

[22] I. Foster, N.T. Karonis, C. Kesselman, G. Koenig, and S. Tuecke, *A Secure Communications Infrastructure for High-Performance Distributed Computing*, In proceedings, 6th International Symposium on High Performance Distributed Computing (HPDC '97), Portland, Oregon, U.S.A., August 1997.

[23] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, *A Security Architecture for Computational Grids*, In proceedings of the 5th ACM Conference on Computer and Communications Security Conference, November 1998, pp. 82–92.

[24] V. Welch et all, *Security for Grid Services*, ANL/MCS-P1024-0203, February 2003. In proceedings, 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), Seattle, Washington, U.S.A., June 2003.

[25] A. Bendahmane, M. Essaaidi, A. El Moussaoui, and A. Younes, *Compromised Resources Tolerance in Grid Computing Systems*, Studies in Computational Intelligence, 2010, Volume 315/2010, Intelligent Distributed Computing IV, pp. 145–154

[26] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, *Reputation Systems*, Communications of the ACM, 43(12) , 2000, pp. 45–48.

[27] F. Azzedin and M. Maheswaran, *Integrating Trust into Grid Resource Management Systems*, In Proceedings of the International Conference of Parallel Processing, 2002.

[28] B. Alunkal, I. Veljkovic, G. von Laszewski, and K. Amin, *Reputation-Based Grid Resource Selection*, In Workshop onAdaptive Grid Middleware, 2003.

[29] S. Kamvar, M. Schlosser, and H. Garcia-Molina, *The Eigen-Trust Algorithm for Reputation Management in P2P Networks*, In Proceedings of the Twelfth International WorldWide Web Conference, 2003.

[30] E. Papalilo, and B. Freisleben, *Managing Behaviour Trust in Grid Computing Environments*, Journal of Information Assurance and Security 1 (2008) pp. 27–37

[31] C. Chen, G. Li-ze, N. Xin-xin, and Y. Yi-xian, *An Approach for Resource Selection and Allocation in Grid Based on Trust Management System*, First International Conference on Future Information Networks, ICFIN 2009, Beijing, pp.232–236.

[32] V. Vijayakumar, and R. S. D. Wahida Banu, *Trust and Reputation Aware Security for Resource Selection in Grid Computing*, IJCSNS International Journal of Computer Science and Network Security, VOL. 8 No. 11, November 2008, pp. 107–115.

Edited by: Costin Bădică and Viorel Negru
Received: December 20th, 2010
Accepted: December 31st, 2010