IADIS International Journal on Computer Science and Information Systems
Vol. 9, No. 1, pp. 55-70
ISSN: 1646-3692

MALICIOUS WORKERS TOLERANCE IN AN AGENT-BASED GRID RESOURCE BROKERING SYSTEM

Naoual Attaoui, Ahmed Bendahmane
Information and Telecommunication Systems Laboratory, Faculty of Science, Tetuan, Morocco
Mohammad Essaaidi
National School of computer science and systems analysis, ENSIAS, Rabat, Morocco
Sugata Sanyal
Corporate Technology Office, Tata Consultancy Services, India
Maria Ganzha, Marcin Paprzycki, Katarzyna Wasielewska
Systems Research Institute Polish Academy of Sciences, Warsaw, Poland

ABSTRACT

The Agent in Grid (*AiG*) project aims to develop an agent-based infrastructure that facilitates the resource management in the Grid. The core assumptions behind the system are: software agents representing Grid resources work in teams, and all information is ontologically represented and semantically processed. Thus far, trust management in *AiG* system is approached from the perspective of breaching an agreement between *Users* and teams or contracts between teams and *Workers*. Unfortunately, it is also possible that some *Workers*, which fulfilled the agreement, could have maliciously returned incorrect results from the jobs that they executed. In this paper, we discuss how the trust management in the *AiG* project can be improved by using a reputation-based voting technique, in order to assure reliable job execution.

KEYWORDS

Grid computing, agent system, trust management, malicious workers tolerance, majority voting, reputation.

1 Introduction

Grid computing can be seen as a way of pooling the capacity of geographically distributed computing resources. It allows completion of large-scale computations, which may not be easily realizable otherwise. Indeed, Grid computing may even allow obtaining computing power above the level available through the use of most powerful supercomputers. A classic example is the SETI@HOME volunteer computing project [SETI@HOME website]. During peak years of its popularity, the total power of computers working on the project was consistently about 10 times larger than the power of a single largest supercomputer of that time.

Computational Grids can be conceptualized as means of realizations of two scenarios. First, the more popular one, when machines combined in a Grid are well defined, bound by formal agreements, administered by administrators, etc. This is the case of a *closed Grid*. Here, issues involved in trust management should not apply, as the Grid is formed on the basis of precisely formulated rules (e.g. formal contracts between known participants). The second situation is similar to that of volunteer computing. This scenario goes back to the origins of Grid computing, where the Grid is conceptualized as an open environment where resources are to be bought and sold in a similar way that the electricity is produced and traded [Foster I. et al., 2002]. In this case *anyone* could join the open (global) Grid to buy and sell computing resources. This second scenario provides the context for this paper. Here, the key characteristics of the Grid is the openness towards the outside world. As a result, large number of *Workers* (resources) of *unspecified* origin participate in computations. As a result, it is possible that *malicious* participants provide false job results to the *Users*. Note that, in the case of volunteer computing systems, such as the BOINC/SETI@HOME, this problem was addressed by a replication-based mechanism, where randomly selected computations are repeated by multiple participants to find those who cheat (see, section 2 for more details). While this approach is simple and relatively effective, it is likely to affect the overall performance, due to the high degree of redundancy.

This being the case, the (global) Grid should have mechanisms to defend itself against malicious *Workers*. Only then, the trust of the *Users*, especially from the business community, can be build (for a survey of *User* trust issues, see [Ermisch J. and Gambetta D., 2006]). These mechanisms have to guarantee authenticity, confidentiality and integrity of results. To achieve this, the "trust" notion needs to be addressed within the open Grid, so that the trustworthiness in such systems can raise their attractiveness for the *Users*.

In this context, an agent-based Grid resource management system was recently proposed – the Agents in Grid (*AiG*) project. Within this project, software agents representing resources work in teams managed by team leaders. The team-based approach was proposed, among others, to reduce disastrous effects of *Worker* disappearance, which has to be taken into account in an open Grid. However, in the *AiG* system, trust management was considered only somewhat superficially, focusing on contractual relations between *Users* and teams, and team leaders and *Workers*. However, the possibility of existence of malicious *Workers* was not considered. Nevertheless, it has to be recognized that there is a need for mechanisms controlling jobs result correctness, to improve the reliability of job execution in the *AiG* system and, consequently, provide a higher level of trust. In other words, we have to introduce sabotage prevention mechanisms that will allow every member of the Grid to make sure that the job results are not affected by unauthorized modifications (see, also [Kumar P.S. et al., 2011]). Therefore, the material presented in this paper extends preliminary investigations presented in [Attaoui N. et al., 2014] for developing methods to improve trust in the *AiG* system.

The remainder of this paper is organized as follows. In the next section we start with analysis of the related work. We follow with an overview of the *AiG* system. Next, we discuss trust management in this system. Finally, we propose a solution to deal with malicious resources.

2 Related work

The problem of ensuring integrity of results of computation has been addressed by many researchers. Most of their results are based on sabotage tolerance techniques like reputation, replication with voting, and credibility-based voting.

Reputation systems [Resnick P. et al., 2000] are commonly used to estimate the reliability of Grid resources, based on history of correctness of results delivered by the *Workers*. In these systems, several entities share information about the trustworthiness of other entities. The cooperation of entities can speed-up the detection of malicious participants. The advantage of this approach is in its simplicity, while its disadvantage is that it is rather primitive, and therefore does not provide the correct picture of the *Workers*' reputation. Specifically, this method has problems fighting against *Workers* that behave well for a long period of time, in order to gain credibility, and after that start to sabotage the results. Furthermore, a reputation system itself can be subject of attacks, where sources try to promote their trustworthiness by making manipulative reports (see [Hoffman K. et al., 2009, Luke Teacy W. T. et al., 2006]). Therefore, even for the reputation systems themselves, mechanisms for detecting untrustworthy resources are required.

A replication-based mechanism, also known as majority voting, is used for ensuring correctness of results (to detect and tolerate erroneous results) by several global computing systems such as BOINC, SETI@home, Folding@home and Mersenne (see [BOINC website], [Cuenca-Acuna F. M. et al., 2003]). This mechanism is based on the idea of replicating each task to multiple *Workers*, to let them perform the same task. Next, the results returned for a task are classified into groups (named result groups) according to the value of the results. These results are verified by the master, using a voting technique, to decide, which result should be accepted as the correct result of the task. The main benefit of the replication approach is its simplicity. On the other hand, its major weakness lies in wasting resources, since to complete a task, several instances need to be computed.

Credibility-based sabotage tolerance approach is another way (proposed in [Sarmenta L. F. G., 2002]) of reducing the acceptance of erroneous results as correct ones, by using together voting and spot-checking techniques. This approach increases the efficiency and automatically guarantees proper balance in the trade-off between performance and correctness. The number of replicas is dynamically determined at the runtime, in accordance with credibility values given to different elements of the system: *Worker*, result, result group, and task. These credibility values are assigned primarily on the basis of the number of spot-checks given to *Workers*, and their past behaviors. In order to check the credibility of *Workers*, the master assigns a spotters task to a *Worker* with probability q (called the spot-check rate). The credibility of a *Worker* is an estimate of the probability that 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, and is equal to the credibility of the *Worker* who returns that result. The credibility of a result group is the conditional probability that this result is correct. Finally, the credibility of a task is defined as the credibility of the group having highest credibility. The system accepts the final results only if the task credibility reaches a certain threshold θ defined by the maximum accepted error for the result $\theta = 1 - \epsilon_{acc}$. Note that this approach can mathematically guarantee that the error rate will not exceed a given acceptable value ϵ_{acc}.

A reputation-based voting technique was recently proposed in [Bendahmane A. et al., 2010]. This technique is an improvement of the credibility-based sabotage tolerance technique in a way that both of them use the majority voting. Here, the investigation of the trustworthiness of workers is based *also* on reputation (not only on credibility). In fact, the reputation formula is developed by introducing the availability and security level parameters, together with the credibility of each *Worker*, to assess efficiently the behavior of such *Worker*. This method of computing reputation can be more robust that the others [Bendahmane A. et al., 2010]. This is why we opted for this solution, especially since the availability is one of key factors in the *AiG* system.

3 Overview of the *Agents in Grid* project

Let us now present a high-level overview of the *Agents in Grid* (*AiG*) system. The *AiG* project attempts at following the ideas originally outlined in [Foster I. et al., 2004] to integrate the Grid and agent systems in order to facilitate the resource management and help *Users* in Grid utilization. The project uses ontologies to represent knowledge. In the system, agents work in teams (see [Kuranowski W. et al., 2008a, Kuranowski W. et al., 2008b]). Each team is managed by its “leader,” the *LMaster* agent. Agent teams utilize services of the Client Information Center (represented by the *CIC* agent) to advertise work they are ready to do (and, possibly, that they are seeking *Worker* possessing specific capabilities). The *CIC* agent plays the role of a central repository where information about all agents “present in the system” is stored [Dominiak M. et al., 2006]. In addition to the *LMaster* and *Workers*, each team has a *LMirror*. This agent stores a copy of the information necessary for the team to persist in case when the *LMaster* crashes. The proposed approach is represented in the form of a Use Case Diagram depicted in Figure 1.

Functionality of the system (in its original form) can be summarized by two scenarios: a *User* who wants to execute a task, and a *Worker User* who wants to join a team (to sell resources and earn money by doing so). When the *User* is seeking a team to execute its job, it specifies job execution constraints to its dedicated *LAgent*. The *LAgent* contacts the *CIC* to obtain the list of teams that can execute its job and utilizes trust information to select teams that can do the job. Next, negotiations between the *LAgent* and the *LMaster*s representing teams ensue and, hopefully, result in reaching an agreement (formalized in the Service Level Agreement; SLA). When the *Worker User* would like to earn money by offering services of its computer(s) (by becoming a *Worker* in a team), it specifies its conditions of joining. The *LAgent* interacts with the *CIC* to obtain a list of teams that are seeking *Workers* satisfying characteristics of hardware and software it represents. Next, it utilizes trust information and the FIPA Contract Net Protocol [Fipa contract net protocol] to establish if a proper team to join can be found.

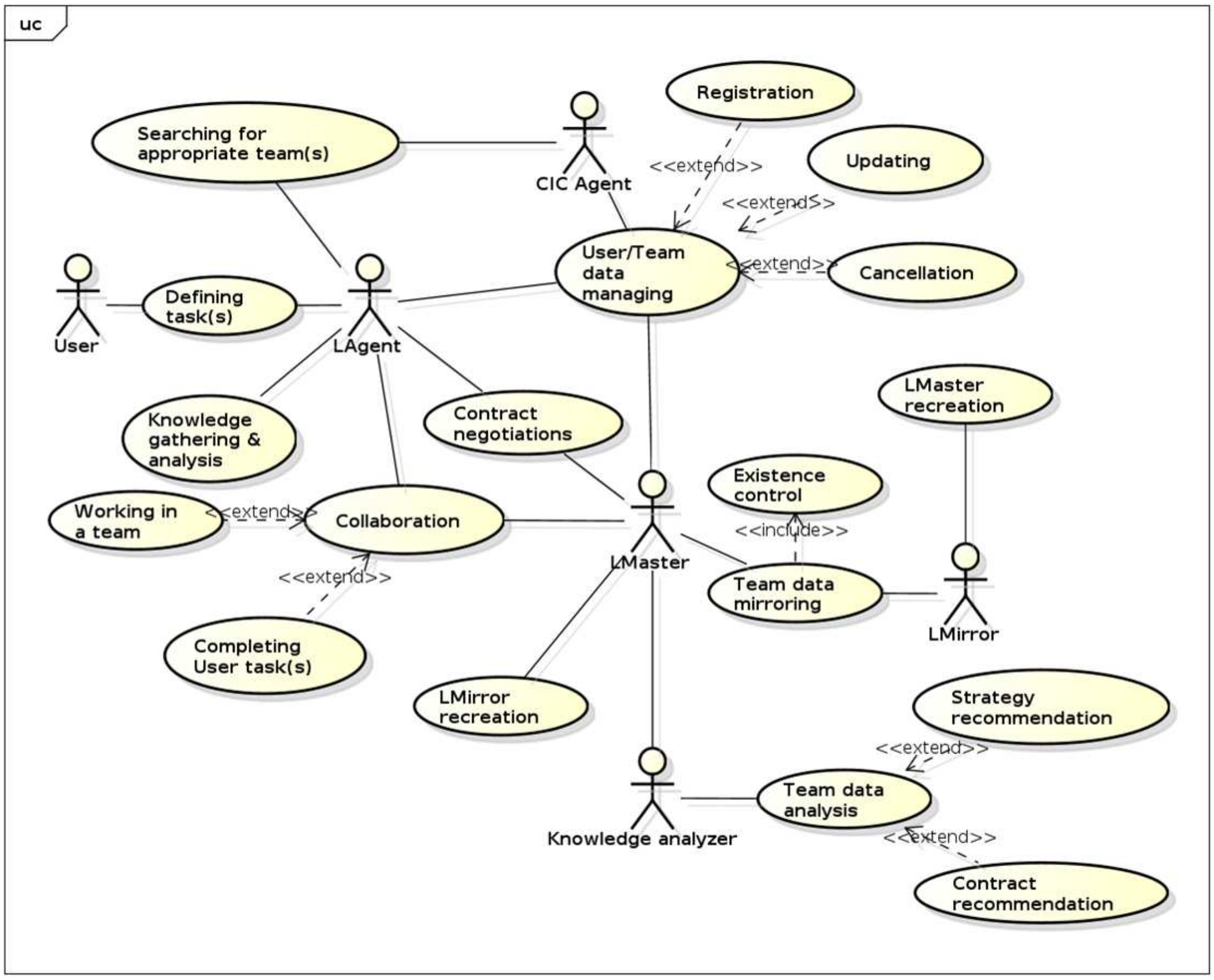


Figure 1: Use Case Diagram of the proposed system.

3.1 Ontologies in the system

Use of ontologies for representation of knowledge has recently attracted attention. It is considered a very promising domain of research for the development of a new generation of information systems. Therefore, it was decided to ontologically model all concepts materializing in the *AiG* system. Thus the needed ontology had to cover all aspects of the structure of the Grid (e.g. characteristics of Grid resources), the negotiation phase and the Service Level Agreement specification.

Our research gave us ideas about the existing Grid ontologies (for an overview, see [Drozdowicz M. et al., 2009]), so we concluded that the closest and the most adequate ontology was the *Core Grid Ontology(CGO)* ([Drozdowicz M. et al., 2009], [Xing W. et al., 2005]). However, some modifications and extensions had to be brought to this solution, especially to include concepts related to the *Service Level Agreement* (*SLA*). The discussion of the adjustments made in the *CGO* ontology can be found in [AiG Ontology]. The complete description of the resulting set of ontologies can be found in [Drozdowicz M. et al., 2010] and in [Drozdowicz M. et al., 2011]. Here, let us briefly outline their main features. The *AiG* ontology hierarchy is structured into three layers (see Figure 2).

The main *AiG* ontologies used in the system are:

1. *AiG* Grid Ontology – directly extending the *CGO* concepts with additional constructs for description of the Grid structure and the resource configuration,

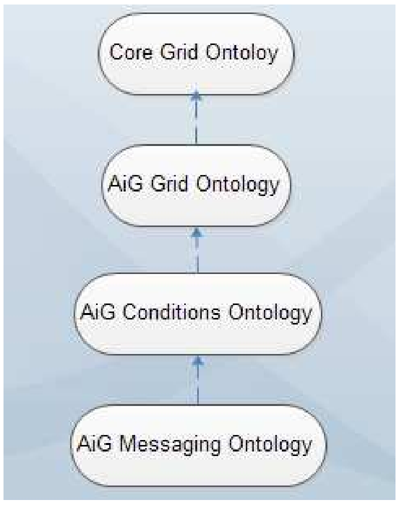


Figure 2: *AiG* ontology hierarchy .

2. *AiG* Conditions Ontology – includes classes and properties required by the SLA negotiations, specifically *Worker* and job execution contract specification (e.g. pricing, payment mechanisms, *Worker* availability conditions, etc.).

3. *AiG* Messaging Ontology – contains definitions of messages, and their possible content, exchanged by the agents, forming the communication protocols of the system.

Note that, as the development of the system progressed further, the ontological structures started to become complex and we were confronted with reasoning problems. Therefore the ontology reengineering became a necessity and was duly completed (for details see [Szmeja P. et al., 2013]).

3.2 Negotiations in the system

In the Grid, obtaining access to the resources, as well as resource management have to be facilitated. In an open Grid (where resources have individual owners) conflicting interests of the *User* and the owner of the resource have to be taken into account. For instance, while the *User* may require certain properties of the resources to be guaranteed (e.g. some QoS), the may wish to maximize her earnings. A typical resolution of this situation is through negotiations, resulting in a Service Level Agreement (SLA). The description of the resource, its characteristics, its constraints, and various guarantees are negotiated by two parties: resource providers and consumers. The SLA can be seen as a contract, defining the commitments of the provider to the quality of service, and describing penalties in case of breaching the agreement. This quality must be measured according to the individual criteria accepted by both parties. Therefore, the SLA contains a set of agreements between the *User* and the provider. Henceforth, autonomous negotiations, and the SLA management, are a key part of the *AiG* system ([Wasielewska K. et al., 2011]).

Possible methods of establishing the SLA contracts are the subject of plentiful literature. During the development of the *AiG* system, after considering European projects integrating business and Grid infrastructures (see, [Parkin M. et al., 2008]), it was decided to base communication on the FIPA Iterated Contract-Net Protocol [FIPA Iterated Contract Net Interaction Protocol], and involve both negotiable (e.g. deadline penalty, job execution timeline), and static parameters (e.g. resource description specified by the *User*).

Currently, a simplified version of the protocol, the FIPA Contract-Net Protocol, is used in the system. However, in the near future its complete multi-round version will be implemented. The difference between the FIPA Contract-Net Protocol and its iterated version is the possibility for the initiator to iterate the process by issuing a revised *Call-For-Proposal*s to a selected subset of participants, while rejecting others. The goal of this protocol is to allow the initiator to seek better offers, by modifying the original *CFP* and requesting new offers.

Note that the *LAgent* negotiates with more than one *LMaster* and therefore the same set of interactions take place in all these negotiations. Let us consider them in the case of a team joining scenario. As a preliminary state, each team manager sends *Worker* acceptance criteria with needed *Worker* configuration to the CIC. Figure 3 represents the process taking place when a *Worker* is attempting to join a team. The *LAgent* representing potential *Worker* registers with the *CIC* (if not registered already). Next, it requests from the *CIC* a list of teams that specified *Worker* acceptance criteria that match resource configuration send by the *LAgent*. As a result, the *CIC* agent responds with a list of *LMaster* agents representing suitable teams. Upon receiving the list of teams the *LAgent* removes from it teams that, according to its current judgment, cannot be trusted. In what follows, the *LAgent* negotiates directly with the *LMaster*s of the remaining teams. In the initial step, the *LAgent* sends out the *Call-For-Proposal* (*CFP*) message to all of them. The *CFP* contains the resource description and the conditions of joining. Upon receiving the *CFP*, to somewhat improve the safety of the system, we envisage that the *LMaster* will contact the *CIC* to make sure that this particular *LAgent* is registered with the system. Here, we assume that only *LAgent*s that are registered with the *CIC* can join agent teams. On the basis of the *CFP*, the *LMaster*s prepare their response. The *CFP*s that do not satisfy hardware / software requirements are refused. Responses from *LMaster*s can have the following forms: (1) refusal (an ACL-REFUSE message), (2) lack of response in a predefined by the *LAgent* time, or (3) a specific offer (an ACL-PROPOSE message). Note that, in a recent paper, we have discussed how the *LMaster*s can generate a pack of offers instead of a single one; thus integrating the multiple equivalent offers strategy into the negotiations ([Attaoui N. et al., 2013]). The *LAgent* awaits for a specific time for the responses and then finds the best of them using a multi-criteria analysis. If the best available offer surpasses its own private valuation, an agent team is selected to be joined. If no acceptable offer is received, the *User* is informed and the *LAgent* awaits further instructions.

According to the Contract-Net Protocol, since the *LAgent* was the originator of the negotiations, it has to be the receiver of the final confirmation. Negotiations concerning job execution have the same structure and flow.

4 Trust management in the *AiG* system

4.1 Preliminary considerations

As argued above, trust is a vital component in Grid computing. For instance, *Users* must trust that the providers will deliver the service they advertise, while the provider must trust the *User* is able to pay for the services used. Otherwise, "honest negotiations" cannot take place. Thus, the Grid infrastructure should protect the *Users* from the owners of the resources and vice-versa. Therefore, in the *AiG* project, we took a preliminary look into basic issues involved in trust relationships in the system. As discussed in [Ganza M. et al., 2007], there exist four situations that are influenced by the trust between *Users* and *LMaster*s (teams) and between *LMaster*s and *Workers*:

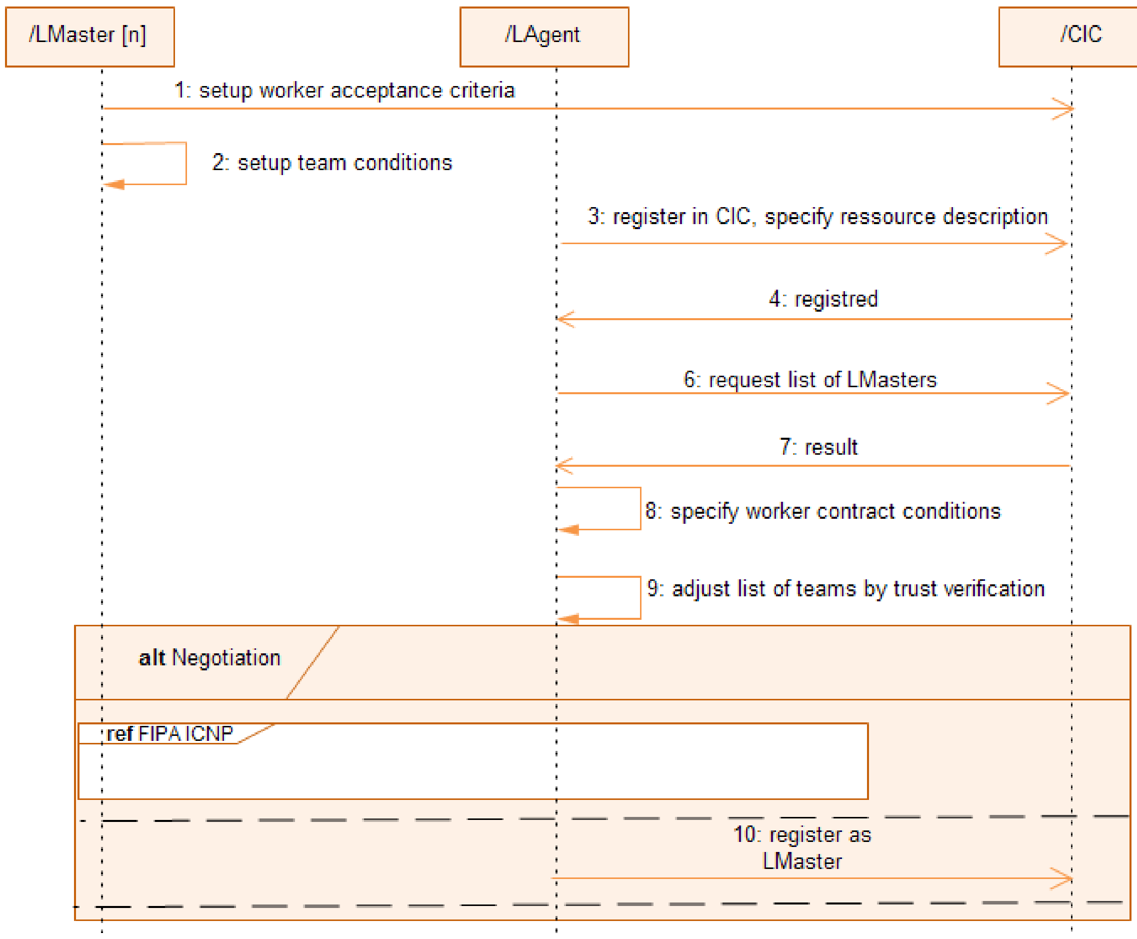
1. When the *LAgent* obtains the list of teams that it can join, it checks if they are trustworthy; for instance, if the *LAgent* worked for a given team and the terms of agreement were not fulfilled (e.g. it was promised that it will be utilized at least 30% of time, but it was not the case), then the *LAgent* may not want to work for that team again.

2. When the *LAgent* obtains the list of teams that can execute its task, it checks if they are trustworthy; for instance, if a given team promised to complete the task within 5 hours and did not, then the *LAgent* may not want to work with such team.

3. When the *LMaster* receives a *Call-For-Proposal* from the *LAgent* that wants to join its team, but this *LAgent* broke an agreement in the past (e.g. it was not available everyday between 10 PM and 6 AM), then it may not want to have such *Worker* in its team.

4. When the *LMaster* receives a *Call-For-Proposal* from the *LAgent* that tried to avoid paying for the last job, then it may not want to get to business with it.

For more details about these four scenarios and to establish how trust materializes in each case and how it can be managed in the system, refer to [Ganza M. et al., 2007].



/LMaster [n]
/LAgent
/CIC
1: setup worker acceptance criteria
2: setup team conditions
3: register in CIC, specify ressource description
4: registred
6: request list of LMasters
7: result
8: specify worker contract conditions
9: adjust list of teams by trust verification
alt Negotiation
ref FIPA ICNP
10: register as LMaster

Figure 3: Sequence diagram for team joining scenario.

4.2 Limitations of trust management in the *AiG* system

Following the discussion presented in [Ganza M. et al., 2007] we assume that, for the time being, trust management in *AiG* system is approached from the perspective of breaching an agreement between resource *Users* and owners, or between team *Workers* and the *LMasters*. For example: when the *LMaster* receives a *Call-For-Proposal* from the *LAgent* that wants to join its team, the *LMaster* checks if it is trustworthy. The main information used for assessment of trust / reliability of a *Worker* is the availability, which is a parameter in the SLA. 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 processed in the Grid against possible malicious behaviors of resources that can return erroneous results. This creates the requirement that open Grid environments should detect and deal with malicious resources, which tamper with the computation and return corrupted results, in order to guarantee reliable execution.

As stated above, in this paper we evaluate the possibility to apply the approach proposed in [Bendahmane A. et al., 2010], for dealing with malicious team *Workers*, by using majority voting mechanisms, and then investigating the trustworthiness of these *Workers*. This approach is based on the reputation management system.

5 Considered solution and its match with the structure of the *AiG* system

The approach evaluated for dealing with malicious *Workers* in *AiG* system is based on reputation to improve the efficiency of majority voting mechanisms ([Bendahmane A. et al., 2010]). Note that, with majority voting it is possible to decide about the trustworthiness of a result immediately after collecting results of all replicas of a task. This approach tolerates a certain number of incorrect results in a vote. However, it fails when, for example, an adversary gets control (through a virus) over a large number of computing resources, and makes them return the same result. In the proposed approach, the decision will be postponed until the moment enough information to infer the trustworthiness of the result is gathered. To collect this information, it has been decided to use existing reputation lists of different computing resources. The key concept in the proposed solution is the combination of credibility, availability, and security, to compute the overall reputation of each Grid resource.

In the next section, we describe the Grid model appropriate to the proposed approach, to see if there are common points with the *AiG* infrastructure. Later, we consider the possibility of integration of the proposed solution with our system, including key technical aspects of such integration.

5.1 Feasibility

In Section 3, we have presented the infrastructure of the *AiG* system. Let us now see what kind of Grid architecture is adequate for the majority voting approach that we consider for adoption. Sabotage tolerance techniques are applied mostly in Grid systems that employ the master-worker computational model [Bendahmane A. et al., 2010]. This Grid model is not restrictive and maps well on the wide variety of Grids. The approach requires a server that distributes work units of an application to Grid nodes. To apply the considered approach the concept of a *Virtual Organization* (*VO*) is used. In Grid computing, a *VO* typically refers to a dynamic set of individuals or institutions defined around a set of resource-sharing rules and conditions. Figure 4 shows the basic components of a Grid system of interest, which consists of N *Virtual Organizations* (*VO*s).

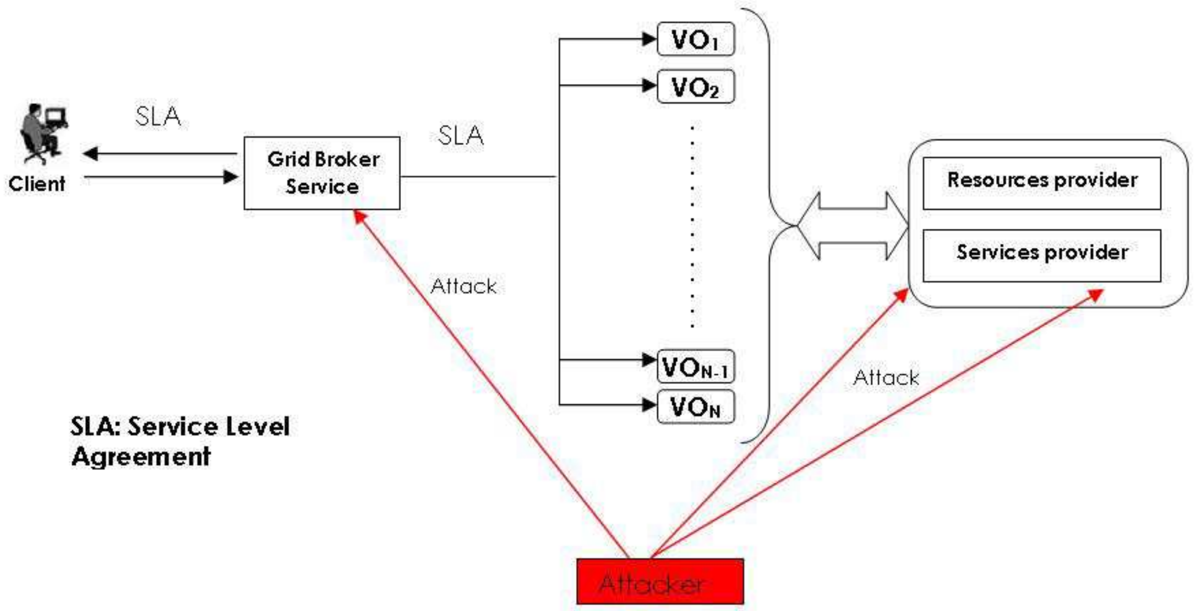


Figure 4: The Grid model appropriate to Sabotage tolerance techniques.

In this environment, after establishment of the SLA, a client submits a job to the Grid broker service. The broker mediates user access to Grid resources, by discovering suitable services and resources provided by *VO*s and deploying and monitoring job execution. On the other hand, each *VO* in the Grid has a single *VO Manager*, which searches for available services. When it finds a relevant 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*.

In this job execution scenario, there are several possible attacks against Grid resources, which might tamper with the computations and return corrupted results (Grid resources takeover by an unauthorized user can result in malicious acts like corruption of job data, job termination, etc.). Such attacks are usually related to security holes of the services and resources provider [Jiancheng N. et al., 2007].

It should be clear that the Grid model proposed for deployment of the sabotage tolerance technique is very similar to the infrastructure of the Grid utilized in the *AiG* system. Note that in the *AiG*, the Grid Broker and the *VO Manager* roles are combined and represented by the *LMaster* and the Grid resources discovery that is facilitated by the *CIC* agent. To clarify things, we can elaborate the analogy between both models, as presented in Table 1:

Model appropriate to Sabotage tolerance techniques	**Model proposed by *AiG* system**
VOs	Teams
VO manager	LMaster
Resources provider	LAgent

Table 1: Analogy between Model appropriate to Sabotage tolerance techniques and Model proposed by *AiG* system

Thus, it seems that the technique in question can be applied to the *AiG* system, in order to bring to it the advantages of robust trust management. Let us now see in more details how the sabotage tolerance techniques can be conceptualized in the *AiG* system.

5.2 Solution

The proposed solution is based on reputation, so let us first see how the *AiG* system can build the reputation value of each *Worker*. In the proposed solution, the reputation of each resource (*Worker*) is built through combining scores valuing its credibility, availability, and security level. In what follows, we specify how these concepts are computed, by taking into account the proposal put forward in [Bendahmane A. et al., 2010], without forgetting the specificity of our system.

5.2.1 Credibility

The credibility is computed by adapting the spot-checking techniques from [Sarmenta L. F. G., 2002]. In the spot-checking, the *LMaster* does not replicate the work that is being completed for the customers, but instead randomly gives a *Worker* a spotter work, correct result of which is already known. This technique can be implemented with or without a black list. Blacklisting allows exclusion of *Workers* that failed the tests.

In [Bendahmane A. et al., 2010], credibility is computed without blacklisting, but in our system it can be very interesting if we have a blacklist stored by the *CIC* (which is assumed to be trustworthy). Then, if a *Worker* is caught giving a bad result, it is immediately blacklisted and will be left out of the team (it can be even removed from the system completely; even though this requires further considerations). Furthermore, when a team receives a *CFP* from that *Worker*, its *LMaster* can immediately refuse this CFP. Specifically, it can ask the *CIC* if such resource has been included in the blacklist or not (see figure 5).

Otherwise, the credibility of a resource C_i, which correctly computed K_i spotters (tasks), when blacklisting is used, will be computed using the following equation [Sarmenta L. F. G., 2002]:

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

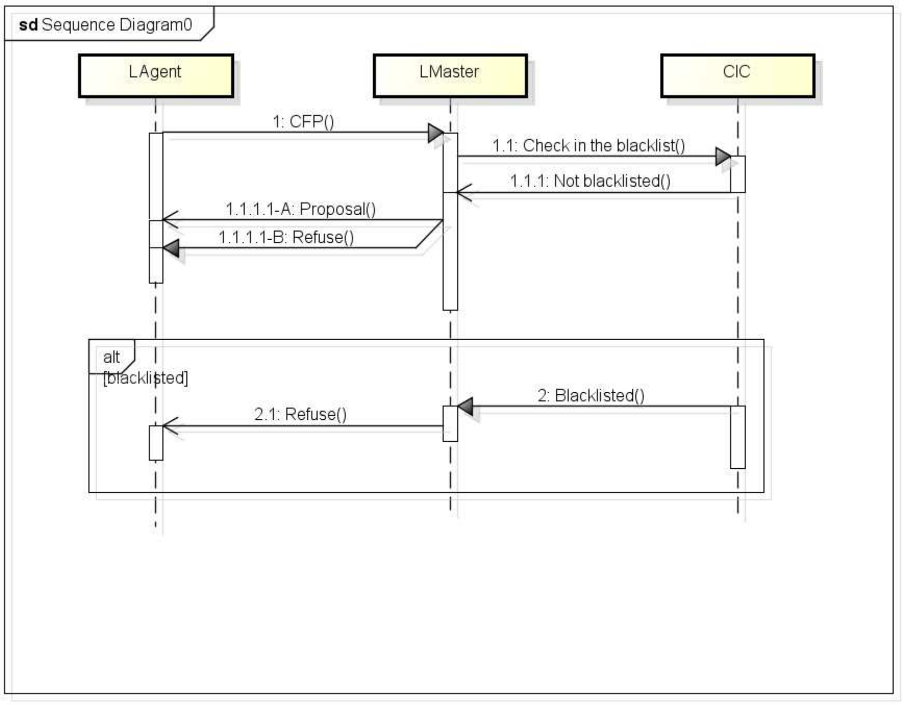


Figure 5: Diagram for checking a *LAgent* in the black list.

Here, f is the proportion of malicious *Workers* who may intentionally submit bad results, e is the base of the natural logarithm and $1 - f$ is the minimal credibility. To calculate f precisely we should use the blacklist that can give us an idea about number of malicious *Workers* in the Grid. But initially, we can start by taking an arbitrary value that will be adjusted in accordance with the number of malicious *Workers* blacklisted.

If a computing resource returns a result validated by the reputation-based majority voting, described in detail in the next section, the credibility value is incremented using the following equation:

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

We consider each successfully validated task a passed spotter. Then, we update the credibility value. In the same manner, we decrement the credibility of those resources whose result was not validated by the reputation-based majority voting using formula:

$$CR(C_i, K_i) = CR(C_i, K_i - 1) \tag{2}$$

After the resources pass enough tasks (K_{min}), they succeed to obtain enough credibility (minimum credibility) for the system to assume that their results are correct.

5.2.2 Availability

Availability is the second parameter that influences the calculation of reputation in the proposed solution (for more details, see section 5.2.4). The availability A_i is the ratio of the number of successful contacts of the resource C_i and the total number of requests. As introduced in [Ganza M. et al., 2007], a monitoring mechanism to check availability of *Workers* (based on responding to test-ping) can be applied here. Henceforth, the computation of availability is calculated as follows:

$$A_i = \frac{N_S}{N_R} \tag{3}$$

Where N_S is the number of successful pings to the resource, and N_R is the total number of pings during the time of the contract.

5.2.3 Security level

The utilized approach considers the security level of a computing resource as a self-protection capability. As defined in [Chen C. et al., 2009], the self-protection capability of a computing resource is calculated by aggregating the security factors like firewall capabilities and anti-virus capabilities. The values of these factors belong to the interval $[0, 1]$. Based on their contributions to security, a proportion is given to each security factor $W(f)$, and aggregated to compute the self-protection capability. For more details see [Bendahmane A. et al., 2010]. The security level is calculated using the following formula:

$$SL = \frac{\sum_{f=1}^{n} W(f)A(f)}{n} \tag{4}$$

Where n is the total number of factors, $W(f)$ is the proportion and $A(f)$ value of the factor.

5.2.4 Team worker reputation

The reputation value R_i of a *Worker* C_i that will be used by the reputation-based majority voting, is the product of the following three metrics: credibility, availability, and security level, and it is computed using the following equation:

$$R_i = CR(C_i, K_i) \times A_i \times SL_i \tag{5}$$

If the result produced by the reputation-based majority voting competition is accepted, the credibility of the resource will be increased (for more details, see section 5.2.1) and thus the reputation of the resource will be also increased. If not, it will be decreased.

5.2.5 Majority voting

As stated, the proposed solution is based on majority voting, and follows primarily [Bendahmane A. et al., 2010]. Overall, the *LMaster* distributes n replicas of a task to several resources C_i , but it may happen that the results will differ (for any reasons), so that it can collect m different results V_j, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$. Notice that $m \leq n$. Each collected result is seen as a vote in a voting pool with n voters. To make a decision, which result V_j is trustworthy (i. e. accepted result), the *LMaster* utilizes a majority voting based on the reputation criteria. This method has a minimum redundancy of 2 [Wong S., 2006], so at least $n = 2$ replicas are needed. Observe that for high values of n we risk causing a slowdown in processing actual jobs. Therefore, while each team may have its own strategy for selection of n, it is suggested that it should be only slightly larger than $n = 2$.

Let us note that all *LAgent*s in the *AiG* system have to be registered with the *CIC*. This means that we could obtain reputation value $R_i \in [0, 1]$ for each *LAgent* by asking the *CIC* (that can store these values). Therefore, when a potential *Worker* approaches a team, its *LMaster* contacts the *CIC* to obtain the combined reputation score, and to use it to decide if that *Worker* is malicious or not (see figure 6).

Let $T(V_j, C_i)$ represent the relationship between the result V_j and the *Worker* C_i. It is calculated by the *LMaster* as follows:

$$T(V_j, C_i) = \begin{cases} 1, & \text{if } C_i \text{ obtained result} V_j \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

We define the resulting reputation $RR(V_j)$ of a given result V_j as the sum of reputations of the *Workers* returning the result V_j. For each result V_j:

$$RR(V_j) = \sum_{i=0}^{n} T(V_j, C_i) \times R_i \tag{7}$$

where R_i is the reputation of *Worker* C_i. To make a decision about the most reliable *Worker*(s), we fix a positive threshold value $\lambda < 1$ and we find the maximum of $RR(V_j)$; $(j = 1, ..., m)$. Here, λ depends on the trust level required by the team. If λ is high the trust level of the Grid is going to be high.

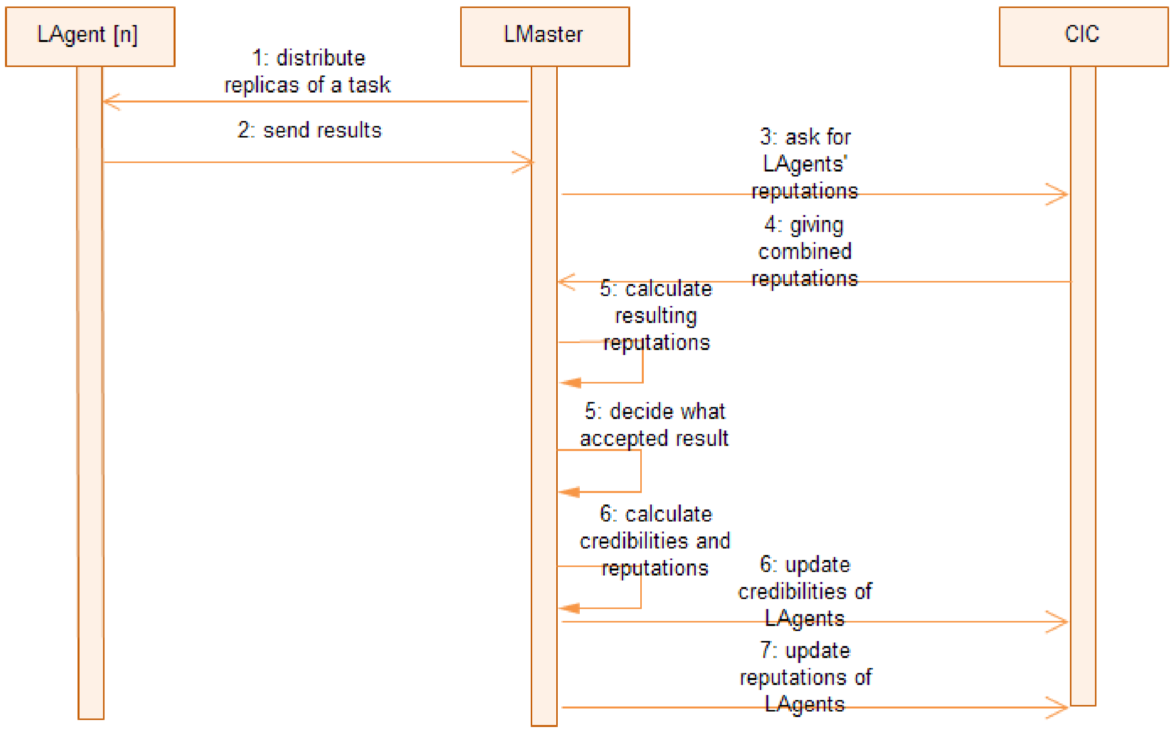

LAgent [n]
LMaster
CIC
1: distribute replicas of a task
2: send results
3: ask for LAgents' reputations
4: giving combined reputations
5: calculate resulting reputations
5: decide what accepted result
6: calculate credibilities and reputations
6: update credibilities of LAgents
7: update reputations of LAgents

Figure 6: Sequence diagram for reputation-based majority voting.

Summarizing, we can judge the correctness of the result V_j as follows:

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

Furthermore, to avoid the possibility that a set of *Workers*, with a low reputation, could undermine the result, we impose the following condition:

$$R_i = \begin{cases} 0, & \text{if } R_i < \theta \\ R_i, & \text{otherwise.} \end{cases}$$

where θ represents the minimum reputation value a *Worker* should have for its results to be taken in consideration.

Observe that, the value of the reputation can be a good criterion of decisions to be made by the *LMaster*, which wants to choose the best *Worker* to become its *LMirror* in case of disappearance of the latter ([Ganza M. et al., 2009]); or for the *LMirror*, that has to create a new *LMaster*. We can also think that every *LMaster* can fix its own minimum of reputation value. So, that the incoming *Worker* that doesn't reach the threshold will not be accepted to become a member of the team.

5.3 Extending the *AiG Ontology* to match the requirements of the proposed solution

From the considerations presented in section 5.2, it can be concluded that one of the crucial aspects of incorporating the proposed solution into the *AiG* system will be the extension of the existing *AiG* ontologies. Here, recall that all information stored and processed in the system is ontology-based (see, Section 3.1). Since concepts related to trust management are distinct from other concepts modeled in the ontologies that have been created thus far, we believe that it is necessary to create a new *AiG Trust Ontology* that would extend the *AiG Grid Ontology*, and would be imported by the *AiG Message Ontology*. Furthermore, the definition of messages exchanged during verification of trustworthiness of system components shall be added to the *AiG Message Ontology*.

Storing trust-related information in the ontology requires modeling concepts related to the blacklist(s), credibility, availability, security level and the overall reputation value. Note that the respective information is stored by the following system components:

- *CIC* – blacklist of *Worker* agents, information about credibility and reputation of *Worker* agents registered in the system,
- *LMaster* – credibility, availability and security level related data for each of *Worker* agents in the team.

More specifically in the *AiG Trust Ontology* the following concepts are going to be distinguished:

- *Blacklisted* – class which instances are on the blacklist in the *CIC*,
- *Credibility* – class which instances represent specific credibility described with properties:
  - *hasCRValue* – float value of calculated credibility of a *Worker* agent,
  - *hasSpotterTests* – number of successfully passed spotter tests,
- *Availability* – class which instances represent specific availability described with properties:
  - *hasAVValue* – float value of calculated availability of a *Worker* agent,
  - *hasPingNumber* – number of ping tests,
  - *hasPongNumber* – number of passed ping tests,
- *Security level* – class which instances containing information about security factors and overall combined security level of a given agent:
  - *hasSLValue* – float value of calculated security level of a *Worker* agent,
  - *hasSecurityFactor* – list of security factors present in the resource configuration,
- *Security Factor* – superclass for specific security factors represented with subclasses e.g. *Firewall*, *Antivirus*, and domain for property *hasSFValue* that indicates the security factors values in a given resource configuration,
- *Reputation* – class representing combined reputation value of a *Worker* agent.

Additionally, a new property *hasReputation* with domain *Grid Component* (concept representing computing or storage component / resource available in Grid) and range *Reputation* has to be included in the ontology. For the *Reputation* the following properties are defined:

- *hasValue* – float value of the calculated reputation based on the values indicated in the *Credibility*, *Availability* and *Security level* properties,
- *hasCredibility* – with the range *Credibility* class,
- *hasAvailability* – with the range *Availability* class,
- *hasSecurityLevel* – with the range *SecurityLevel* class.

Note, that the *LMaster* agent will store ontology with instances for its team members and all defined trust-related properties, besides the blacklist information. On the other hand, the *CIC* will store the ontology with all *Worker* agents registered in the system and defined properties regarding only their reputation and credibility values as well as the blacklist information.

Finally, the *AiG Message Ontology* will now include the following new message definitions:

- *BlacklistCheckRequest* – message of type ACL-REQUEST send from the *LMaster* to the *CIC* with the content ontology listing agents that should be verified,
- *BlacklistCheckResponse* – message of type ACL-INFORM send from the *CIC* to the *LMaster* with the content ontology listing agents that were found on the blacklist,
- *ReputationRequest* – message of type ACL-REQUEST send from the *LMaster* to the *CIC* with the content ontology listing agents, which reputation has to be checked,
- *ReputationResponse* – message of type ACL-INFORM send from the *CIC* to the *LMaster* with the content ontology listing agents with filled property *hasReputation*,
- *UpdateTrustInformation* – message of type ACL-INFORM send from the *LMaster* to the *CIC* with the content ontology listing agent and their actual credibility and reputation values.

6 Concluding remarks

The aim of this paper was to discuss issues involved in trust management in an agent-based Grid resource management system. We have been convinced that in an open Grid environment, we have to deal with the following situation: some malicious *Workers* are interested in corrupting the results of a job. In this context, a recently published paper introduced a new approach to improve trust in open Grids. The proposed approach utilized a majority voting mechanism improved by a reputation management system. This approach was analyzed as a possible solution for trust management in the *AiG* system, to make it more robust and to improve its credibility. Furthermore, the reputation calculated by this approach seems to be very useful in the negotiation with incoming *Workers*, as well as in the re-creation of *LMaster*s or *LMirror*s. Since the initial analysis of the proposed approach was quite promising, we have considered technical details of its potential implementation, including creation of an additional ontology. This ontology is to model all concepts necessary or robust trust management in the *AiG* system.

Future studies related to the trust management will discuss the idea of informing each party in the contract, how trustworthy is the other party, before starting negotiations, by associating a trust factor to each entity in the *AiG* system. The calculation of trust factor will be inspired by the TSET [Borgohain R. et al., 2012], which is an improvement over the existing Secure Electronic Transaction (SET) protocol [Li Y. and Wang Y., 2012]. Trust factor of an entity will be decided by the total number of contracts the entity is involved in and the total number of contracts that it brokered in the past. So, before negotiating, each entity can check the trust factor of the other entity from the *CIC* and decide itself whether it wants to participate in the contract or not. We will report on our progress in subsequent publications.

References

[Attaoui N. et al., 2013] *Multiple equivalent simultaneous offers strategy in an agent-based grid resource brokering system - initial considerations*. Scalable Computing: Practice and Experience. Vol. 14, No. 2, pp. 8394.

[Attaoui N. et al., 2014] *Malicious Workers Tolerance in an Agent-Based Grid Resource Brokering System - Preliminary Considerations*. Proceedings of the The 2014 Int'l Conference on Computational Science and Computational Intelligence (CSCI'14), in press.

[Bendahmane A. et al., 2010] *Reputation-Based Majority Voting For Malicious Grid Resources Tolerance*. Scalable Computing: Practice and Experience, Vol. 11, No. 4, pp. 385392.

[Foster I. et al., 2002] *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Open Grid Service Infrastructure WG, Global Grid Forum.

[Ganza M. et al., 2007] *Trust Management in an Agent-based Grid Resource Brokering System-Preliminary Considerations*. AIP Conference Proceedings, Vol. 946, No. 35, pp. 35-46.

[Kumar P.S. et al., 2011] *Recent Trust Models In Grid*. Journal of Theoretical and Applied Information Technology, Vol. 26 No. 1, pp. 64-68.

[Ermisch J. and Gambetta D., 2006] *Peoples trust: The design of a survey-based experiment.* ISER Working Paper Series, Discussion Paper No. 2216.

[Fipa contract net protocol] http://www.fipa.org/specs/fipa00029/SC00029H.html.

[Sarmenta L. F. G., 2002] *Sabotage-tolerance mechanisms for volunteer computing systems.* Future Generation Computer Systems, Vol. 18, No. 4, pp. 561-572.

[Jiancheng N. et al., 2007] *Threat analysis and Prevention for grid and web security services.* In Proc. of Eighth ACIS. International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 526531.

[Chen C. et al., 2009] *An Approach for Resource Selection and Allocation in Grid Based on Trust Management System*, First International Conference on Future Information Networks, pp. 232236.

[Foster I. et al., 2004] *Brain meets brawn: Why grid and agents need each other.* Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Vol. 1, pp. 8-15.

[Kuranowski W. et al., 2008a] *Forming and managing agent teams acting as resource brokers in the grid-preliminary considerations.* International Journal of Computational Intelligence Research, Vol. 4, No. 1, pp. 9-16.

[Kuranowski W. et al., 2008b] *Super- vising agent team an agent-based grid resource brokering system-initial solution.* Proceedings of the Conference on Complex, Intelligent and Software Intensive Systems, pp. 321-326.

[Drozdowicz M. et al., 2009] *Ontologies, Agents and the Grid: An Overview.* in B.H.V. Topping, P. Ivnyi, (Editors), "Parallel, Distributed and Grid Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, Chapter 7, pp 117-140.

[Core grid ontology] http://grid.ucy.ac.cy/grisen/cgo.owl.

[Xing W. et al., 2005] *Design and Development of a Core Grid Ontology.* In Proc. of the CoreGRID Workshop: Integrated research in Grid Computing, pp 21-31.

[Drozdowicz M. et al., 2010] *Utilization of Modified Core GRID Ontology in an Agent-based Grid Resource Management System.* Proceedings of the CATA 2010 Conference.

[Drozdowicz M. et al., 2011] *Trends in parallel, distributed, grid and cloud computing for engineering.* Chapter Ontology for Contract Negotiations in Agent-based Grid Resource Management System. Saxe-Coburg Publications, Stirlingshire, UK.

[AiG Ontology] http://gridagents.svn.sourceforge.net/viewvc/gridagents/trunk/ontology/AiGOntology/.

[Wasielewska K. et al., 2011] *Negotiations in an Agent-based Grid Resource Brokering Systems.* in P. Ivnyi, B.H.V Topping,(Editors), "Trends in in Parallel, Distributed, Grid and Cloud Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK.

[Parkin M. et al., 2008] *A Comparison of SLA Use in Six of the European Commissions FP6 Projects.* Institute on Resource Management and Scheduling, CoreGRID-Network of Excellence, pp. 1-38.

[Dominiak M. et al., 2006] *Efficient Matchmaking in an Agent-based Grid Resource Brokering System.* Proceedings of the International Multiconference on Computer Science and Information Technology pp. 327335.

[FIPA Iterated Contract Net Interaction Protocol] http://www.fipa.org/specs/fipa00030/index.html

[Resnick P. et al., 2000] *Reputation Systems.* Communications of the ACM, Vol. 43, No. 12, pp. 4548.

[Hoffman K. et al., 2009] *A survey of attack and defense techniques for reputation systems.* ACM Computing Surveys, Vol. 42, No. 1, pp. 1-1.

[Luke Teacy W. T. et al., 2006] *Travos: Trust and reputation in the context of inaccurate information sources.* Autonomous Agents and Multi-Agent Systems, Vol. 12, No. 2, pp. 183198.

[BOINC website] http://boinc.berkeley.edu

[SETI@HOME website] http://setiathome.berkeley.edu

[Cuenca-Acuna F. M. et al., 2003] *Autonomous Replication for High Availability in Unstructured P2P systems.* The 22nd International Symposium on Reliable Distributed Systems, pp. 99-108.

[Zuev Y.A, 1998] *On the estimation of efficiency of voting procedures.* Theory of Probability Its Applications, Vol. 42, No. 1, 7181, pp. 73-81.

[Borgohain R. et al., 2012] *TSET: Token based Secure Electronic Transaction.* International Journal of Computer Applications 45(5): pp. 1-6.

[Li Y. and Wang Y., 2012] *Secure Electronic Transaction (SET protocol).* [online] Available at http://www.people.dsv.su.se/ matei/courses/IK2001$_SJE/li-wang_SET.pdf(Accessed15February,2012)$.

[Ganza M. et al., 2009] *Mirroring information within an agent-team-based intelligent Grid middleware; an overview and directions for system development.* Scalable Computing: Practice and Experience, Vol. 10, No. 4, pp. 397-411.

[Szmeja P. et al., 2013] *Reengineering and extending the Agents in Grid Ontology.* Large Scale Scientific Computing LNCS, Springer Germany.

[Wong S., 2006] *An authentication protocol in web-computing.* In 20th Intl. Parallel and Distributed Processing Symp.(IPDPS 2006), Proceedings, Greece. IEEE.