

# AGORA: An Architecture for Strategyproof Computing in Grids

Daniel Grosu

Department of Computer Science

Wayne State University

Detroit, Michigan 48202, USA

Email: dgrosu@cs.wayne.edu

**Abstract**—Grids enable the sharing of resources and problem solving in multi-organizational environments. The problem of resource management in such systems is very complex because the participants, resource owners and users, have their own requirements and objectives that need to be considered when making allocation decisions. To address this issue novel protocols that take into account the self-interest and incentives of these participants need to be developed. These kinds of protocols in which the participants maximize their own utilities only if they report their true parameters and follow the rules are called strategyproof protocols.

In this paper we propose AGORA, an architecture for strategyproof computing in grids and present several strategyproof mechanisms for resource allocation that can be deployed on this architecture.

*Index Terms* — mechanism design, grid computing, resource management, strategyproof mechanism.

## I. INTRODUCTION

Grid computing is a promising new technology that enables the integration of several geographically distributed resources such as computers, instruments, and data storage devices into a coherent computing infrastructure. Several applications in science, engineering and commerce benefited from using the grid technologies [12]. Globus Toolkit [2], in use at hundreds of sites worldwide, emerged as the standard software infrastructure for grids. Globus Toolkit 3 (GT3) is an open source implementation of the Open Grid Services Infrastructure (OGSI) [24] compliant with the Open Grid Services Architecture (OGSA) [13], [14]. GT3 provides a framework for defining a variety of portable services for resource management, data transfer, information discovery and security.

Fundamental to the success of grid computing is the resource management component which is responsible for resource discovery, selection, allocation and monitorization [17]. One of the difficulties of developing efficient resource allocation mechanisms in grids is the fact that the participants, resource owners and users, have their own requirements and objectives that need to be considered when making allocation decisions. These participants have no a-priori motivation for cooperation and they are tempted to manipulate the resource allocation protocol if it is beneficial to do so. This behavior may lead to poor system performance and inefficiency. Unlike the traditional protocols in distributed computing, the new protocols must deal with the possible manipulations. Also,

the system must provide incentives to agents to participate in the given algorithm. The solution of these kinds of problems comes from economics, more precisely from *mechanism design theory* [22]. The scope of this theory is to provide tools and methods to design protocols for self interested agents. Of interest are the so called *strategyproof mechanisms* in which the participants maximize their own utilities only if they report their true parameters and follow the given algorithm. In a mechanism each participant has a privately known function called *valuation* which quantifies the agent benefit or loss. Payments are designed and used to motivate the participants to report their true valuations. The goal of each participant is to maximize the sum of her valuation and payment. As an example consider several resource providers that offer computer services. We assume that each resource is characterized by its job processing rate. An allocation mechanism is strategyproof if a resource owner maximizes her utility only by reporting the true resource processing rate to the mechanism. The optimal utility is independent of the values reported by the other participating resource owners.

The *strategyproof computing* paradigm proposed in [18] considers the self-interest and incentives of participants in distributed computing systems. The core of a strategyproof system is an open market for mechanisms in which participants can deploy locally strategyproof mechanisms. These kinds of mechanisms satisfy the incentive properties within their local scope.

In this paper we consider the strategyproof computing paradigm in the context of grid computing. We develop an architecture for strategyproof computing in grids. The core of this architecture is a set of local markets for mechanisms. Two types of mechanisms can be deployed in a local market for mechanisms. The first type are the mechanisms deployed by users in which the participants are the resource owners. The resource owners submit bids for jobs and according to these bids the user mechanism decides where a job or a set of jobs are sent for execution. The second type are the mechanisms implemented by the resource owners. The users submit bids for computational units and according to these bids the resource mechanism decides what user will send jobs for execution. Each user participating in a resource mechanism and respectively each resource owner participating in a user mechanism has an associated broker. The broker is responsible

for mechanism discovery, mechanism selection, bid and job submission.

### Related work

Applications of economic principles to resource management in distributed computing have been investigated by several researchers [1], [6], [7], [8], [19], [25], [27]. In [25] an extendable architecture (GREED) that supports several economic models and enables the integration of these models into the existing Globus Toolkit is presented. Auctioning based resource allocation is studied in [7]. Wolski *et al.* [27] proposed a computational economy for controlling resource allocation in grids and studied the efficiency of auctioning and commodity market models. Abramson *et al.* [1] proposed a service oriented grid computing system called Nimrod-G. A comprehensive survey on economic models for resource management can be found in [5].

Recently, several researchers considered the mechanism design theory to solve several computational problems that involve self interested agents. These problems include resource allocation and task scheduling [20], [26], [27], routing [9] and multicast transmission [10]. Nisan and Ronen [21] were the first to consider the mechanism design problem in a computational setting. They studied different types of mechanisms for shortest path and job scheduling on unrelated machines. They proposed a VCG(Vickrey-Clarke-Groves) mechanism for solving the shortest path in graphs where edges belong to self interested agents and mechanisms for solving the task scheduling on unrelated machines. The VCG mechanism allows arbitrary form for valuations and is restricted to objective functions defined by the sum of agents' valuations. Archer and Tardos [3] derived a general framework for designing truthful mechanisms for optimization problems where the private data of each agent is described by one real valued parameter. Their method allows the design of strategyproof mechanisms for optimization problems that have general objective functions and restricted form for valuations. Using this method they designed strategyproof mechanisms for several problems in computer science. In [4] the same authors investigated the frugality of shortest path mechanisms. A strategyproof mechanism that gives the overall optimal solution for the static load balancing problem in distributed systems is proposed in [15]. Ng *et al.* [19] proposed a strategyproof system for dynamic resource allocation in data staging. The results and the challenges of designing distributed mechanisms are surveyed in [11].

### Our contributions

We develop AGORA, a grid computing architecture based on the strategyproof computing paradigm. We present several strategyproof mechanisms for resource allocation that can be deployed on our architecture. We define these mechanisms and describe their properties in the context of AGORA. We simulate and study the implementation of one resource allocation mechanism on our architecture.

### Organization

The paper is structured as follows. In Section 2 we present AGORA, an architecture for strategyproof computing in grids. In Section 3 we present and discuss several mechanisms that

can be deployed on our architecture. In Section 4 we study by simulation one resource allocation mechanism. In Section 5 we draw conclusions and present future directions.

## II. AGORA ARCHITECTURE

The deployment of strategyproof resource allocation mechanisms in grids involves the development of a suitable architecture. The architecture should consider the existing grid infrastructure provided by the Globus technology and should provide services for mechanism deployment, discovery, validation and certification.

In this architecture grid users can choose to deploy their own mechanisms or to participate in a mechanism deployed by grid service providers. Similarly grid service providers can choose to participate in a user mechanism or to deploy their own mechanisms. This gives more flexibility to the participants to accommodate their requirements and expectations. The mechanisms are deployed in a Local Market for Mechanisms. By having multiple markets for mechanisms the system becomes more scalable. This is necessary because in general the allocation mechanisms have a centralized structure that will not scale well when we increase the number of the participants. Moreover, it will not be possible to deploy a single mechanism for all the grid users.

In Figure 1 we present AGORA, an architecture for strategyproof computing in Grids. The main components of this architecture are:

- Users and user brokers.
- Grid service providers (GSP) and GSP brokers.
- Local grid market for mechanisms (LMM).
- Validation and certification authority (VCA).

There are two types of interactions between users and resource providers depending on who initiates the trading of resources. In the first type Grid Service Providers request bids from the users and select the winning users according to their strategyproof mechanism. In the second type User Brokers requests bids from the Grid Service Providers and select the providers according to a user mechanism. The first type of interactions is handled by Resource Mechanisms (RM) and the second one by the User Mechanisms (UM).

### A. Components involved in Resource Mechanisms

We present the components of AGORA that participate in the first type of interactions handled by the Resource Mechanisms. These are User Brokers (UB), Grid Service Providers (GSP), Local Market for Mechanisms (LMM) and Validation and Certification Authority (VCA). Each GSP posts its mechanism on one of the Local Markets for Mechanisms. Before these mechanisms can be used they need to be validated and certified by the validation and certification authority.

1) *User broker (UB)*: The User Broker is responsible for mechanism discovery, mechanism analysis and selection, bid submission, sending user jobs to resources, collecting the results and providing the user with a uniform view of grid resources. There are four components of the user broker:

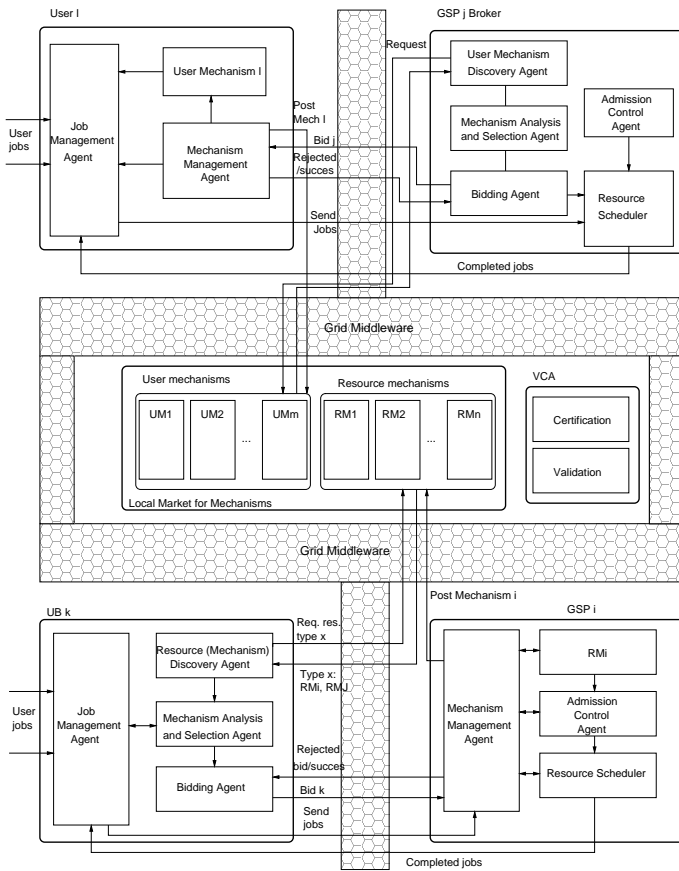


Fig. 1. AGORA architecture.

- **Job Management Agent:** It is responsible for user interaction, job creation, submission and monitorization. It also coordinates the mechanism analysis and selection, resource discovery and the bidding process. When the jobs complete it collects the results of the computation.
- **Resource (Mechanism) Discovery Agent:** It is responsible for resource/mechanism discovery. It sends a request for resources/mechanisms to the Local Market for Mechanisms. The Local Market for Mechanisms sends back the information on the mechanisms that match the request.
- **Mechanism Analysis and Selection Agent:** It is responsible for analyzing the mechanism information submitted by the Local Market for Mechanisms. Based on the user requirements and on the properties of the mechanisms it selects a mechanism in which the user will participate.
- **Bidding Agent:** It is responsible for choosing and submitting the bid to the selected mechanism. If it is a successful bid the Job Control agent sends the user jobs for execution to the GSP that runs the mechanism.

2) **Grid Service Providers (GSP):** GSPs contribute their resources to the Grid and charge the users for services. GSPs create strategyproof mechanisms that are posted on the Local Market for Mechanisms. The Mechanism Management component is responsible for posting the GSP's strategyproof

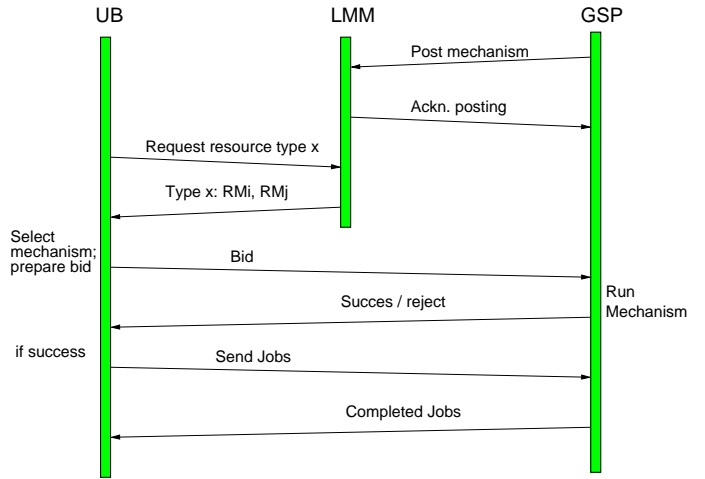


Fig. 2. Protocol for resource mechanisms

mechanism on the LMM. It also runs the allocation mechanism, collects the bids from users and determine the winning users. The winning users are the users that send jobs for execution. Once the winning users are determined it informs the users of the result of the mechanism by sending success or reject messages. It also coordinates the admission control and resource scheduling.

3) **Local Market for Mechanisms (LMM):** It provides support for users and GSPs to deploy strategyproof mechanisms, and enables the users and GSPs to find the right mechanism that match their requirements and preferences. There are two types of strategyproof mechanisms that are deployed in LMM: user mechanisms (UM) and resource mechanisms (RM). LMM takes a request from a user or GSP specified in an appropriate language and returns the type of mechanisms that are requested. LMM must be provided by the grid infrastructure as a component of the grid middleware. LMM should provide an interface and a language for specifying the requests for mechanisms. User brokers and GSPs must submit their requests for mechanisms expressed in this language.

4) **Validation and Certification Authority (VCA):** It provides the validation of the properties of the mechanisms (strategyproof, statistical). After a mechanism is validated by the authority a certificate is issued for that mechanism. This will certify to the participants in the mechanism that the mechanism is strategyproof. To maintain the validity of the certificates, VCA must check periodically the posted mechanisms for their incentive properties. The validation techniques are crucial for maintaining the strategyproof property of the system. The issues and challenges of mechanisms validation are explored in [18].

In Figure 2 we present the protocol that implements the interactions among UB, LMM and GSP for a resource mechanism. The GSP designs a mechanism and posts the mechanism on LMM. Once posted, the mechanism is validated and a certificate is issued by the VCA. UB submits a request for mechanism message to LMM containing the user's requirements described in a specific language. LMM replies to UB

with information about the mechanisms that match the request. UB selects a mechanism, prepares and submits the bid. GSP collects all the bids from the participating users and runs the mechanism to determine the winning users. After the winning users are determined GSP sends success or reject messages to the users to inform them about the outcome of the mechanism. If UB receives a success message it sends its jobs to the GSP that run the mechanism. Jobs are executed and the results of the computation are sent to the user.

### B. Components involved in User Mechanisms

AGORA allows the users to deploy their own allocation mechanisms. Each user will post the mechanism on one of the Local Markets for Mechanisms. Similarly to the previous type of interactions we have four components involved in a user mechanism: User, GSP Broker, LMM and VCA. In this section we present only the User and GSP Broker, the two other components, LMM and VCA were already presented in the previous section.

1) *GSP Broker*: It is responsible for mechanism discovery, analysis and selection, bid submission, admission control, resource scheduling and job execution. There are five components of the GSP broker.

- *User Mechanism Discovery Agent*: It is responsible for handling the interaction with GSP and user mechanism discovery. It sends requests for user mechanisms to the LMM and receives the information about the suitable mechanisms in LMM.
- *Mechanism Selection and Analysis Agent*: It is responsible for analyzing the user mechanisms information submitted by LMM. Based on this information and on the GSP's requirements it selects a user mechanism in which the GSP will participate.
- *Bidding Agent*: It is responsible for choosing and submitting the bid to the selected user mechanism. If it is a successful bid GSP will receive jobs for execution from the user that run the mechanism.
- *Admission Control Agent*: It implements the admission control rules for accepting dynamically-arriving jobs.
- *Resource Scheduler*: It is responsible for resource scheduling and allocation. When the jobs are completed it sends back the results of the computation.

2) *User*: The user runs the mechanism and decides what resources will execute the jobs. The Job Management Agent is responsible for user interaction, job submission and results collection. The Mechanism Management Agent coordinates the user mechanism, posts the mechanism, receives bids, runs the mechanisms and determines the winners. It also informs the participating GSPs of the result of the mechanism by sending success or reject messages.

The protocol that describes the interactions involved in user mechanisms is similar to the protocol presented in Figure 2.

### C. Integration into Globus framework

Globus toolkit provides mechanisms and interfaces for the design and deployment of higher level services based on

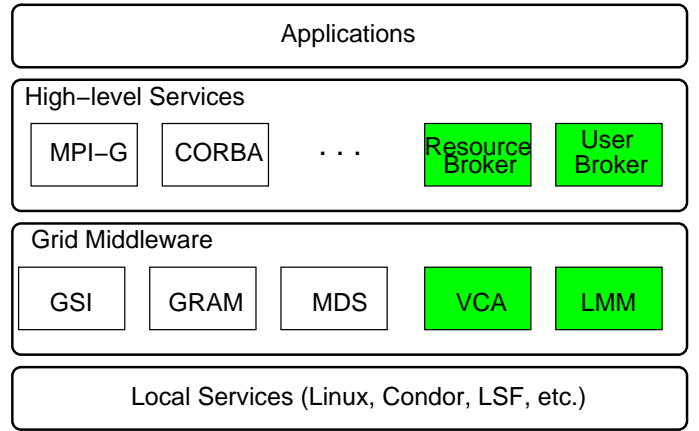


Fig. 3. Integration into Globus framework

a set of low-level service components. The components of the toolkit are: the GT core, Grid Security Infrastructure (GSI), Data Management, Resource Management (GRAM) and Information Services (MDS). GRAM provides an API and a protocol for requesting and using resources.

To provide support for strategyproof computing on the Globus framework two new modules are added, LMM and VCA. LMM module provides a uniform interface and common protocols for mechanism deployment and discovery. VCA module provides mechanisms validation and certification services. Higher level services provided by the user brokers and GSP brokers are based on the services provided by LMM and VCA in addition to the services provided by the existing Globus modules. In Figure 3 we show the integration of the new modules into the Globus framework.

## III. STRATEGYPROOF MECHANISMS

In this section we introduce the main concepts of mechanism design and present examples of user and resource mechanisms that can be deployed on AGORA.

### A. Generalities on mechanism design

We limit our description to mechanism design problems for one parameter agents. In this type of mechanism design problems each agent has some private data represented by a single real valued parameter [3]. In the following we define such problem.

A *mechanism design problem for one parameter agents* is characterized by:

- A finite set  $\Lambda$  of allowed outputs. The output is a vector  $\lambda(\mathbf{b}) = (\lambda_1(\mathbf{b}), \lambda_2(\mathbf{b}), \dots, \lambda_n(\mathbf{b}))$ ,  $\lambda(\mathbf{b}) \in \Lambda$ , computed according to the agents' bids,  $\mathbf{b} = (b_1, b_2, \dots, b_n)$ . Here,  $b_i$  is the value (bid) reported by agent  $i$  to the mechanism.
- Each agent  $i$ , ( $i = 1, \dots, n$ ), has a privately known parameter  $t_i$  called her *true value*. The *valuation* of each agent depends on the output and on her true value and is denoted as  $V_i(t_i, \lambda(\mathbf{b}))$ .
- Each agent goal is to maximize her *utility*. The utility of agent  $i$  is  $U_i(t_i, \mathbf{b}) = P_i(\mathbf{b}) + V_i(t_i, \lambda(\mathbf{b}))$ , where  $P_i$  is the payment handed by the mechanism to agent  $i$ .

(iv) The goal of the mechanism is to select an output  $\lambda$  that optimizes a given cost function  $g(t, \lambda)$ .

*Definition 3.1: (Mechanism)* A mechanism is characterized by two functions:

- (i) The *output function*  $\lambda(\mathbf{b}) = (\lambda_1(\mathbf{b}), \lambda_2(\mathbf{b}), \dots, \lambda_n(\mathbf{b}))$ . This function has as input the vector of agents' bids  $\mathbf{b} = (b_1, b_2, \dots, b_n)$  and returns an output  $\lambda \in \Lambda$ .
- (ii) The *payment function*  $P(\mathbf{b}) = (P_1(\mathbf{b}), P_2(\mathbf{b}), \dots, P_n(\mathbf{b}))$  that gives the payment handed by the mechanism to each agent.

*Notation:* In the rest of the paper we denote by  $\mathbf{b}_{-i}$  the vector of bids not including the bid of agent  $i$ . The vector  $\mathbf{b}$  is represented as  $(\mathbf{b}_{-i}, b_i)$ .

*Definition 3.2: (Strategyproof mechanism)* A mechanism is called *strategyproof* if for every agent  $i$  of type  $t_i$  and for every bids  $\mathbf{b}_{-i}$  of the other agents, the agent's utility is maximized when she declares her real type  $t_i$  (i.e. truth-telling is a dominant strategy).

A desirable property of a mechanism is that the utility of a truthful agent is always non-negative. The agents hope for a profit by participating in the mechanism.

*Definition 3.3: (Voluntary participation mechanism)* We say that a mechanism satisfies the *voluntary participation condition* if  $U_i(t_i, (\mathbf{b}_{-i}, t_i)) \geq 0$  for every agent  $i$ , true values  $t_i$ , and other agents' bids  $\mathbf{b}_{-i}$  (i.e. truthful agents never incur a loss).

## B. User Mechanisms

1) *Minimum Job Execution Time Mechanism:* We assume that the user that runs the mechanism has a contiguous sequence of jobs that need to be executed. The jobs are generated by the user at the rate  $\Phi$ . Each GSP $_i$  participating in the mechanism is characterized by the *true value*  $t_i$  represented by the inverse of its processing rate,  $t_i = \frac{1}{\mu_i}$ . Only GSP $_i$  knows  $t_i$ . The mechanism asks each GSP $_i$  to report its value  $b_i$  (the inverse of its processing rate). The GSPs may not report the true value. After all the GSPs report their values the mechanism computes an output function (i.e. the loads assigned to computers),  $\lambda(\mathbf{b}) = (\lambda_1(\mathbf{b}), \lambda_2(\mathbf{b}), \dots, \lambda_n(\mathbf{b}))$ ,  $(\sum_i^n \lambda_i(\mathbf{b}) = \Phi)$  according to the GSPs' bids such that the overall expected job execution time is minimized and hands a payment  $P_i(\mathbf{b})$  to each GSP. All GSPs know the mechanism and the algorithm used to compute the output function (allocation).

GSP $_i$ 's ( $i = 1, 2, \dots, n$ ) valuation is defined as:

$$V_i(t_i, \lambda(\mathbf{b})) = -t_i \lambda_i(\mathbf{b}) \quad (1)$$

which is equivalent to GSP $_i$ 's utilization. The greater the utilization, the smaller the valuation. We assume each GSP wants to choose its strategy (what value  $b_i$  to report) such that its utility is maximized. The utility for each GSP is defined as the payment received from the mechanism plus the valuation determined by the number of jobs allocated to it:

$$U_i(t_i, \mathbf{b}) = P_i(\mathbf{b}) + V_i(t_i, \lambda(\mathbf{b})) \quad (2)$$

The goal is to design a strategyproof mechanism that minimizes the overall expected response time of the system composed of the  $n$  GSPs. This involves finding an allocation algorithm and a payment scheme that minimizes the overall expected response time according to the GSP bids  $b_i$  and motivates all the GSPs to bid their true values  $t_i$ . For our mechanism we use the optimal allocation algorithm (OPTIM) presented in [23]. Using this algorithm and the method presented in [3] we designed a strategyproof mechanism that minimizes the overall job execution time [15]. In the following we define the mechanism.

*Definition 3.4: (Minimum Execution Time Mechanism)* The Minimum Execution Time Mechanism is defined by the following two functions:

- (i) The allocation function given by the OPTIM algorithm.
- (ii) The payment function is given by:

$$P_i(\mathbf{b}_{-i}, b_i) = b_i \lambda_i(\mathbf{b}_{-i}, b_i) + \int_{b_i}^{\infty} \lambda_i(\mathbf{b}_{-i}, x) dx \quad (3)$$

This mechanism is a strategyproof mechanism that satisfies the voluntary participation condition. The proofs of the properties of the mechanism are given in [15].

Because the optimal algorithm assumes a central dispatcher, the mechanism will be implemented in a centralized way as part of the user code. In the following we present the protocol that implements the minimum job execution time mechanism (MJET).

### Protocol MJET:

After the mechanism is posted on LMM and it is validated, it is ready to receive bids from the GSPs. GSP $_i$  replies with bid  $b_i$  to the mechanism.

- 1) After the user mechanism collects all the bids it does the following:
  - 1.1. Computes the allocation using OPTIM algorithm.
  - 1.2. Computes the payments  $P_i$  for each GSP using equation (3).
  - 1.3. Sends  $P_i$  to each GSP $_i$ .
- 2) Each GSP receives its payment and evaluates its profit.

GSPs will obtain the maximum profit only when they report the true value.

2) *A mechanism with verification:* Now we present an allocation mechanism in which the payments are sent to the participating GSPs only after the jobs are executed. This is a mechanism with verification and works as follows. It first asks the GSPs to report their processing rates. Having obtained these rates, the mechanism computes the allocation and allocates the jobs to GSPs. Because it is a mechanism with verification the payment to each GSP is computed and given to it after the assigned jobs were executed. Here we assume that the processing rate with which the jobs were actually executed is known to the mechanism. Each GSP's goal is to report a value for its processing rate such that its utility is maximized. The mechanism must be designed such that the

GSPs maximize their utility only if they report the true values of the processing rates and execute the jobs using their full processing capacity. Also, if each GSP reports its true value and executes the jobs at the full capacity then the minimum total latency is obtained.

We consider that a set of  $n$  GSPs participate in the mechanism. We assume that each GSP is characterized by a load-dependent *latency function*. The latency function of GSP $_i$  is linear on  $x_i$  and has the following form:

$$l_i(x_i) = a_i x_i \quad (4)$$

where  $x_i$  is the arrival rate of jobs allocated to GSP $_i$  and  $a_i$  is a parameter inversely proportional to the processing rate of GSP $_i$ . A small (big)  $a_i$  characterizes a fast (slow) resource. In other words  $l_i(x_i)$  measures the time required to complete one job at GSP $_i$ .

We assume that the user that runs the mechanism has a large number of jobs that need to be executed. These jobs are generated by the user at the rate  $R$ . A feasible allocation of jobs to the participating GSPs is a vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  that satisfies two conditions:

- (i) Positivity:  $x_i > 0 \quad i = 1, 2, \dots, n$ ;
- (ii) Conservation:  $\sum_{i=1}^n x_i = R$ ;

The performance of the system composed of the participating GSPs is characterized by the *total latency*:

$$L(\mathbf{x}) = \sum_{i=1}^n x_i l_i(x_i) = \sum_{i=1}^n a_i x_i^2 \quad (5)$$

Given the job arrival rate  $R$  at the system with  $n$  participating GSPs, we want to find a feasible allocation  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  that minimizes the total latency  $L(\mathbf{x})$ .

The latency functions at each GSP are linear, thus the allocation of jobs in proportion to the processing rate of each GSP gives the minimum total latency. The optimal allocation is given by:

$$x_i = \frac{\frac{1}{a_i}}{\sum_{k=1}^n \frac{1}{a_k}} R \quad i = 1, 2, \dots, n \quad (6)$$

which gives the minimum value for the total latency:

$$L^* = \frac{R^2}{\sum_{k=1}^n \frac{1}{a_k}} \quad (7)$$

We call the algorithm that computes this allocation the PR algorithm. The true value  $t_i$  corresponds to the parameter  $a_i$  in the latency function  $l_i$ , which is inversely proportional to the processing rate of GSP $_i$ . GSP $_i$  may execute the jobs at a different rate from the rate reported. We denote the parameter corresponding to this rate by  $\tilde{t}_i$  and call it *execution value*. We want a mechanism that obtains the optimal allocation and forces the participants to reveal their true values  $t_i$  and follow the PR algorithm. In the following we define the mechanism with verification.

*Definition 3.5: (The allocation mechanism with verification)* The mechanism with verification is defined by the following two functions:

- (i) The allocation function given by the PR algorithm.
- (ii) The payment function is given by:

$$P_i(\mathbf{b}, \tilde{\mathbf{t}}) = C_i(\mathbf{b}, \tilde{\mathbf{t}}) + B_i(\mathbf{b}, \tilde{\mathbf{t}}) \quad (8)$$

where the function  $C_i(\mathbf{b}, \tilde{\mathbf{t}}) = \tilde{t}_i x_i^2(\mathbf{b})$  is called the *compensation* function for GSP $_i$ ; and the function  $B_i(\mathbf{b}, \tilde{\mathbf{t}}) = L_{-i}(\mathbf{x}(\mathbf{b}_{-i}, \mathbf{b}_{-i})) - L(\mathbf{x}(\mathbf{b}), (\mathbf{b}_{-i}, \tilde{t}_i))$  is called the *bonus* for GSP $_i$ . The function  $L_{-i}(\mathbf{x}(\mathbf{b}_{-i}, \mathbf{b}_{-i}))$  is the optimal latency when GSP $_i$  is not used in the allocation. Thus, the bonus for a GSP is equal to its contribution in reducing the total latency.

This is a strategyproof mechanism with verification that satisfies the voluntary participation condition. We proved these properties in [16]. An informal description of the centralized protocol implementing the mechanism is as follows. The mechanism collects the bids from each GSP, computes the allocation using PR algorithm and allocates the jobs. Then it waits for the allocated jobs to be executed. In this waiting period the mechanism estimates the actual job processing rate at each GSP and use it to determine the execution value  $\tilde{t}_i$ . After the allocated jobs are completed the mechanism computes the payments and sends them to the GSPs. After receiving the payment each GSP evaluates its utility.

### C. Resource Mechanisms

1) *Vickrey Auction:* We assume that the GSP runs the mechanism (auction) and  $n$  users have chosen to participate in the mechanism. The users have several jobs ready for execution. User  $i$  ( $i = 1, 2, \dots, n$ ) has a private valuation  $t_i$  associated with the execution of her jobs. The mechanism works as follows. Each user sends a bid to the GSP running the mechanism expressing the amount she is willing to pay for her job execution at the GSP. The user who submits the highest bid wins the auction and she submits her job to the GSP for execution. The winning user will pay the price equal to the second highest bid submitted by the users. If there are more than one winning user, the winner is chosen at random from the highest bidding users.

The utility of user  $i$  has the following form:

$$U_i(t_i, b) = \begin{cases} t_i - b_s & \text{if user } i \text{ wins} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

In the above equation,  $b_s$  is the value of the second highest bid and represents the payment given to the GSP by the user. A participating user will maximize her utility only if she bids her true valuation. It is important to mention that the bids are secret and the users who did not win the auction do not know who placed the highest bid and the value of the bid.

## IV. EXPERIMENTAL RESULTS

In this section we study by simulation the resource allocation mechanism with verification (user mechanism). Due to space limitations we consider only this mechanism for our simulation study. We designed a simulator that allows us to execute resource allocation mechanisms on a simulated AGORA architecture. The simulated grid environment we

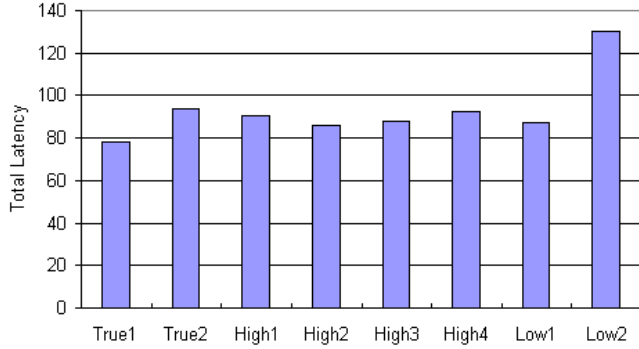


Fig. 4. Performance degradation.

consider consists of 16 GSPs, characterized by the parameters  $a_i$  (proportional to the GSP's processing rate) given in Table I.

TABLE I  
GSP PARAMETERS.

GSP	1 - 2	3 - 5	6 - 10	11 - 16
True value ( $t$ )	1	2	5	10

To study our mechanism we consider eight types of experiments depending on the bid and on the execution value of GSP1. In all the experiments we assume that all the GSPs except GSP1 bid their true values and that their execution values are the same as their true values. We classify these experiments in three main classes according to the bids of GSP1 as follows: *True*, when  $b_1 = t_1$ ; *High*, when  $b_1 > t_1$ ; and *Low*, when  $b_1 < t_1$ . We further divide these main classes considering the execution value of GSP1.

First, we consider the influence of false bids ( $b_1 \neq t_1$ ) on the total latency. We assume that the job rate of one user is  $R = 20$  jobs/sec. In Figure 4 we present the total latency for the eight experiments. We now discuss the results of each experiment.

*True1*: ( $t_1 = t_1 = b_1 = 1$ ) All the GSPs report their true values. The execution value is equal to the true value for each GSP. As expected (from the theory) we obtain the minimum value for the total latency ( $L = 78.43$ ).

*True2*: ( $\tilde{t}_1 > t_1 = b_1, t_1 = 1, b_1 = 1, \tilde{t}_1 = 3$ ) The parameters are as in *True1* except that GSP1 has a higher execution value  $\tilde{t}_1 > t_1$ . This means GSP1 execution is slower increasing the total latency by 17%.

*High1*: ( $\tilde{t}_1 = b_1 > t_1, t_1 = 1, b_1 = 3, \tilde{t}_1 = 3$ ) In this case GSP1 bids three times higher than its true value and the execution value is equal to the bid. Because GSP1 bids higher the other GSPs are overloaded thus increasing the total latency. GSP1 gets fewer jobs and executes them with a slower execution rate.

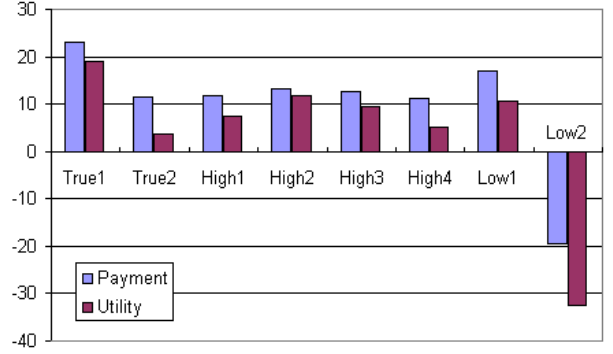


Fig. 5. Payment and utility for GSP1.

*High2*: ( $b_1 > t_1 = \tilde{t}_1, t_1 = 1, b_1 = 3, \tilde{t}_1 = 1$ ) In this case GSP1 gets fewer jobs and the other GSPs are overloaded. Here the increase is not as big as in *High1* because GSP1 executes the jobs at its full capacity.

*High3*: ( $b_1 > \tilde{t}_1 > t_1, t_1 = 1, b_1 = 3, \tilde{t}_1 = 2$ ) This case is similar to *High1* except that the execution on GSP1 is faster. It can be seen that the total latency is less than in the case *High1*.

*High4*: ( $\tilde{t}_1 > b_1 > t_1, t_1 = 1, b_1 = 3, \tilde{t}_1 = 4$ ) Similar to *High1*, except that GSP1 executes the jobs slower, increasing the total latency.

*Low1*: ( $\tilde{t}_1 = t_1 > b_1, t_1 = 1, b_1 = 0.5, \tilde{t}_1 = 1$ ) GSP1 bids 2 times less than its true value, getting more jobs and executing them at its full capacity. The increase in total latency is not big, about 11%.

*Low2*: ( $\tilde{t}_1 > t_1 > b_1, t_1 = 1, b_1 = 0.5, \tilde{t}_1 = 2$ ) In this case GSP1 gets more jobs and executes them two times slower. This overloading of GSP1 leads to a significant increase in the total latency (about 66%).

From the results of these experiments, it can be observed that small deviations from the true value and from the execution value of only one GSP may lead to large values of the total latency. We expect even larger increase if more than one GSP does not report its true value and does not use its full processing capacity. The necessity of a mechanism that forces the participants to be truthful becomes vital in such situations.

In Figure 5 we present the payment and utility of GSP1 for each set of experiments. As expected, GSP1 obtains the highest utility in the experiment *True1*, when it bids its real value and uses its full processing capacity. In the other experiments GSP1 is penalized for lying and the payment that it receives is lower than in the case of *True1*. The utility is also lower in these experiments. An interesting situation occurs in the experiment *Low2* where the payment and utility of GSP1 are negative. This can be explained as follows. GSP1 bids two times less than its true value and the PR algorithm allocates more jobs to it. In addition, its execution value  $\tilde{t}_1$  is two times higher than  $t_1$ , that means the allocated jobs are executed two

times slower than in the experiment *True1* (when  $\tilde{t}_1 = t_1$ ). More allocated jobs to GSP1 combined with a slow execution increases the total latency  $L$ . The total latency becomes greater than the latency  $L_{-1}$  obtained when GSP1 is not used in the allocation (i.e.  $L > L_{-1}$ ) and thus the bonus is negative. The absolute value of the bonus is greater than the compensation and from the definition of the payment it can be seen that the payment given to GSP1 is negative.

## V. CONCLUSION

We have proposed AGORA, an architecture for strategyproof computing in grids that allows grid users and grid service providers to deploy strategyproof mechanisms for resource allocation. We showed how this architecture integrates in the existing grid framework provided by the Globus toolkit. Examples of strategyproof mechanisms deployable on this architecture were presented. Future work will address the implementation of AGORA and the development of suitable techniques for mechanism validation.

## REFERENCES

- [1] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computing Systems*, 18(8):1061–1074, October 2002.
- [2] Globus Alliance. <http://www.globus.org>.
- [3] A. Archer and E. Tardos. Truthful mechanism for one-parameter agents. In *Proc. of the 42nd IEEE Symp. on Foundations of Computer Science*, pages 482–491, October 2001.
- [4] A. Archer and E. Tardos. Frugal path mechanisms. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 991–999, January 2002.
- [5] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource allocation and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.
- [6] K. M. Chao, R. Anane, J. H. Chen, and R. Gatward. Negotiating agents in a market-oriented grid. In *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 406–407, May 2002.
- [7] C. Chen, M. Maheswaran, and M. Toulouse. Supporting co-allocation in an auctioning-based resource allocator for grid systems. In *Proc. of the 11th IEEE Heterogeneous Computing Workshop*, April 2002.
- [8] C. Ernemann, V. Hamscher, and R. Yahyapour. Economic scheduling in grid computing. In *Proc. of the 8-th Intl. Workshop on Job Scheduling Strategies for Parallel Processing*, pages 128–152, July 2002.
- [9] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proc. of the 21st ACM Symp. on Principles of Distributed Computing*, pages 173–182, July 2002.
- [10] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, August 2001.
- [11] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proc. of the 6th ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, September 2002.
- [12] I. Foster and C. Kesselman. *The Grid: a Blueprint for a New Computing Infrastructure*. 2nd edition, Morgan Kaufmann, San Francisco, CA, 2003.
- [13] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Globus Project, June 2002.
- [14] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer Magazine*, 35(6):37–46, June 2002.
- [15] D. Grosu and A. T. Chronopoulos. Algorithmic mechanism design for load balancing in distributed systems. In *Proc. of the 4th IEEE Intl. Conf. on Cluster Computing*, pages 445–450, September 2002.
- [16] D. Grosu and A. T. Chronopoulos. A load balancing mechanism with verification. In *Proc. of the 17th IEEE Intl. Parallel and Distributed Processing Symp. Workshop on Advances in Parallel and Distributed Computational Models*, pages 163–170, April 2003.
- [17] J. Nabrzyski, J. M. Schopf, and J. Weglarz. *Grid Resource Management, State of the Art and Future Trends*. Kluwer Academic Publishers, Boston, MA, 2003.
- [18] C. Ng, D. Parkes, and M. Seltzer. Strategyproof computing: Systems infrastructures for self-interested parties. In *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.
- [19] C. Ng, D. Parkes, and M. Seltzer. Virtual worlds: Fast and strategyproof auctions for dynamic resource allocation. In *Proc. of the ACM Conference on Electronic Commerce*, pages 238–239, June 2003.
- [20] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over Internet - The POPCORN project. In *Proc. of the 18th IEEE International Conference on Distributed Computing Systems*, pages 592–601, May 1998.
- [21] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behaviour*, 35(1/2):166–196, April 2001.
- [22] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, Mass., 1994.
- [23] X. Tang and S. T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In *Proc. of the Intl. Conf. on Parallel Processing*, pages 373–382, August 2000.
- [24] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. *Open Grid Services Infrastructure (OGSI), Version 1.0*. Global Grid Forum, June 2003.
- [25] S. Vazhkudai and G. von Laszewski. A greedy grid - the grid economic engine directive. In *Proc. of the 15th IEEE International Parallel and Distributed Processing Symposium*, April 2001.
- [26] W. E. Walsh, M. P. Wellman, P. R. Wurman, and J. K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Proc. of the 18th IEEE International Conference on Distributed Computing Systems*, pages 612–621, May 1998.
- [27] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: market formulations controlling resource allocation on the computational grid. In *Proc. of the 15th IEEE International Parallel and Distributed Processing Symposium*, April 2001.