Flexible Architecture for Supporting Auctions in Grids

Pablo Chacin[a], Liviu Joita[b], Björn Schnizler[c], Felix Freitag[a]
[a]*Computer Architecture Department, Polytechnic University of Catalonia, Spain*
{pchacin,felix}@ac.upc.edu
[b]*School of Computer Science, Cardiff University, Wales, UK*
l.joita@cs.cardiff.ac.uk
[c]*Information Systems and Management, University of Karlsruhe, Germany*
schnizler@iw.uni-karlsruhe.de

Abstract

Efficient and flexible resource allocation is one of the key factors for a wide application of Grids in business and scientific areas. Recently, the use of auctions for scheduling and allocating Grid resources has been proposed in literature. Hitherto, however, only few of the proposed mechanisms are integrated in Grid infrastructures. Furthermore, none of these proposals are applied in commercial settings. One reason for this lays in technical challenges such as replacing discovery and matchmaking mechanisms by auctions have to be overcome in order to integrate an auction mechanism into a Grid platform. Another barrier for a wide deployment of Grid auctions is that an all-embracing auction mechanism may not exist. A technical and economic sound architecture thus has to support the simultaneous deployment of multiple different auction instances.

The paper introduces a flexible economic Grid middleware which abstracts from the underlying technical infrastructure and supports the simultaneous instances and management of a wide spectrum of different auction. The design concepts of such system are outlined and an ongoing prototype is presented, using the integration of a MACE auction as a case study.

1. Introduction

The increasing development of Grid middleware during the past couple of years, such as Globus Toolkit 4[1], enables the emergence of loosely coupled computing environments that use open, general-purpose protocols in order to deliver nontrivial qualities of services [1]. This paved the way for sharing distributed heterogeneous resources over geographically dispersed organizations. This sharing mechanism has major implications for organizations since it can reduce costs by outsourcing nonessential elements of their IT infrastructure to various forms of service providers. Such e-Utilities – providers offering on-demand access to computing resources – enable organizations to perform computational jobs spontaneously using other resources in the Grid that are not under the control of the (temporary) user [2].

However, the deployment of large-scaled Grid infrastructures is still hampered by several barriers. One of the key issues is to determine which resources are allocated to which applications and scheduled to what time. Most Grid infrastructures employ optimization algorithms, which allocate and schedule resources based on static system specific cost functions [3]. From an economic point of view these algorithms are harmful, as they strive to maximize a system-wide performance objective rather than maximizing an aggregate value. These algorithms can thus not guarantee that buyers who value a resource will really receive it.

The application of auction mechanisms for the Grid is considered to work well. Auctions can provide an efficient allocation of distributed resources. By assigning utilities to their resource requests, users can express their relative needs or costs to the resource, which is subject to service usage constraints. If auctions are properly designed and integrated into existing Grid middleware, users may be provided with incentives to express their true values for resource requests and offers. Thus, an efficient allocation of computational resources can be achieved.

The application of auctions to the resource allocation mechanism in Grids is not a new approach. In literature, several mechanisms are proposed, such as Vickrey auctions [4], continuous double auctions [5], or combinatorial mechanisms [6]. The multitude of proposed auction types for different application scenarios suggests that an all-embracing auction mechanism for the Grid may not exist. This is caused by the fact that application

---

[1] See http://www.globus.org/ for details

scenarios are too specific to be covered by a single mechanism.

Users prefer to select a mechanism due to the rules this particular mechanism offers. For instance, suppose a user wants to purchase a computational service and wants to negotiate over the speed of the underlying processor, the main memory, and the storage space. In this case the user may select a multi-attribute auction, but not a single-attribute auction. A second example: suppose a user wants to bid for a dedicated machine with standard configuration requirements, having a high probability of receiving a suitable machine. This user may select a high liquid market running a continuous double auction which offers a market order. These two examples show that auction types are chosen according to the requirements of the market participants.

Furthermore, it is important to consider that each concrete organization that is providing resources to a Grid could have specific policies for its resources. One of the main challenges of a Grid allocation mechanism is to coordinate the resource usage policies established by the participant organizations. A single instance of an auction which is capable of handling such diversity of policies seems impractical (if possible at all). Instead, it would be desirable to let each participant select an auction mechanism that best matches its policy requirements or even setup its own auctions if none is satisfactory.

The contribution of this paper is to propose an architecture for the simultaneous instantiation of different auction types on a Grid infrastructure to allow the resource trading according to the participant's needs. This architecture is independent of the specific auction mechanisms being used and the grid platform on which it is implemented.

The paper is structured as follows: Section 2 identifies requirements upon a technical sound Grid market middleware on the base of a scenario. Subsequently, section 3 outlines a market middleware which fulfils these specified requirements and presents the implementation of a MACE auction as a case study. Section 4 discusses the considerations of implementing grid auctions in realistic scenarios. Section 5 reviews related literature. Finally, section 6 concludes the paper and gives an outlook on future research.

2. Requirement Analysis

The design of an architecture which supports the application of auctions in the Grid has to meet both technical and economic requirements. Technical requirements reflect Grid middleware specific issues such as the integration into existing toolkits. Economic requirements are related to the support of a multitude of different auction mechanisms. In the following, the requirements upon the economic Grid architecture are derived from a scenario.

Scenario

Let us suppose a scientific workgroup which develops a physical simulator within an university. Several external computational entities are required in order to perform the desired experiments. For instance, computational, storage, and database services as well as a specialized simulator application and some physical measurement devices are required by the scientists. After the simulation is performed, data and results have to be analyzed using statistic services and afterwards stored on tapes.

For purchasing required services, the scientists firstly want to discover the suitable and instantiated auctions in the network. For the required computational - the database and the storage service, they discover an instance of a multi-attribute combinatorial exchange. As these three types of services are common services in the market, the exchange is liquid for these services. Therefore it can be assumed that the scientists receive immediately their services via the market mechanism. Furthermore, they need to ensure that specific computational and policy characteristics are met. These requirements can also be satisfied by the exchange, as it supports commodities with multiple attributes. As such, the scientists use this mechanism to bid for the computational, storage, and database service.

Subsequently, they consider using the exchange for purchasing the required simulation service and the physical devices. However, after they have analyzed the order book of the exchange, they recognize that none of these (more specialized) services are offered in this auction. If they would send a bid to the market to requesting these services, they may run the risk of being executed at an unfavourable price. The reason is that an exchange is usually favourable for high liquid markets. For low liquid markets, the exchange may result in inefficient allocations. As such, the scientists decide to use an English auction instead of the exchange. An English auction is advantageous for specialized services with a low liquidity.

Having purchased the required services in order to perform their experiments, the scientists require the statistic and tape services for evaluation and storing their results. However, they do not discover any suitable auction instances for these services. Thus, they decide setting up their own auction and propagate these auctions within the network. As the scientists are not familiar with existing Grid middleware, they want the technical complexity to be hidden when setting up their auction. The scientists decide to setup a single-sided reverse auction as this auction type is advantageous in terms of low auction durations. After they performed these two auctions, they have purchased all required services and can start their experiment.

Requirement Specification

Based upon this scenario and a survey of related Grid literature the following ad-hoc requirements upon the architecture are elicited:

Requirement 1: Integration into existing Grid middleware

A suitable economic architecture for the Grid should be flexible enough to be integrated in different Grid middleware systems such as Unicore[2] or Globus Toolkit 4. Auction based allocation are used to complement and enhance, instead of substitute, existing grid platforms. Interoperability is key to assure that existing grid infrastructure and do not impose restrictions to participants to join the market.

Requirement 2: *Isolation of the auction mechanism to the Grid middleware*

The architecture should hide the complexities of the underlying Grid middleware from the auction mechanisms. Auction should work despite the specific platforms used by participants and be limited to handle the specific issues of establishing and running auctions. Interfaces must be provided so that auction instances can access information about resources and handle its actual allocation to auction winner.

Requirement 3: *Support for different auction types*

Several different auction types such as single-sided and double-sided auctions with varying parameter and configurations should be supported by the architecture.

Requirement 4: Dynamic *Auction Discovery and Join.*

The architecture has to support the dynamic discovery of running auction instances, without requiring a centralized registry or the centralization of the administration. Participants should be able to seamlessly join multiple existing auctions that meets its requirements.

Requirement 5: Dynamic *Setup of auction instances*

A participant must be able to setup, on the fly a new instance of an auction mechanism, which should immediately available to other participants. This way any participant can decide to establish ad-hoc, per-resource, auctions to buy very precise, one shot requirements or sell rare resources. More common resources and repeating requirements can be satisfied using stable, long running auction instances.

Requirement 6: *Coexistence of several auction instances*

The architecture should support the coexistence of several auction instances running in parallel. For instance, it should be possible to run a double auction instance and an English auction simultaneously. The setup of new auction instance should not interfere with existing one, neither should require any centralize management or administrative coordination. This way, the setup of auction instances becomes a purely local task.

3. Grid Market Middleware

The requirements analysed in the previous section require an innovative approach for the construction of a resource allocation middleware. The proposed approach is a framework that offers a set of generic negotiation mechanisms, on which specialized strategies and policies can be dynamically plugged to. The architectures adapts to specific application domains and different market designs. In this section we will first analyze the main concepts of the architecture. Afterwards, we detail, by means of a use case, how different auctions can be implemented. Finally, we will introduce some security concerns.

3.1 Architecture

The Grid Market Middleware (GMM) proposed architecture offers an agent based framework for dynamic location and management of Grid services based on economic criteria. It provides mechanisms to locate and manage registered resources, services, and applications. It locates other trading agents, engages agents in negotiations, learns and adapts to changing conditions. Furthermore, the middleware offers a set of generic negotiation mechanisms, on which specialized strategies and policies can be dynamically plugged in. The service and resource exchanges occur between parties that might join the market in an ad-hoc or opportunistic way. Therefore, the infrastructure which is required to joining the market is minimized. Figure 1 illustrates how the Grid is modelled in terms of clients, service and resource providers.

---

[2] See http://www.unicore.org/ for details.

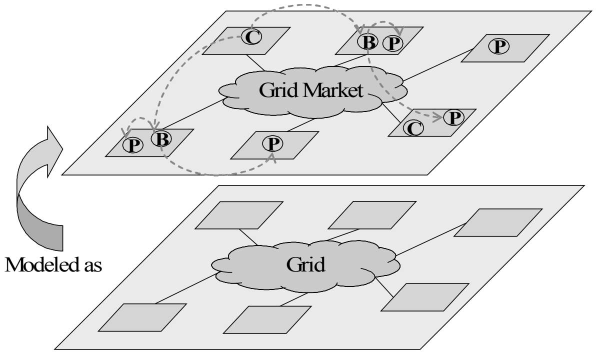

Figure 1: Decentralized allocation in a Grid Market.

The resource allocation middleware has been envisioned as a set of economic agents representing the Grid services consumer applications, the service brokers and Grid services providers, which interact among each others and with the software components of the underlying Grid. The middleware thus coordinates in a decentralized way and uses economic criteria for the assignment of resources.

The middleware – as shown figure 2 – has a layered architecture. The *Application layer* is for the domain specific end user application, such as: collaboration tools, problem solving environments and many others.

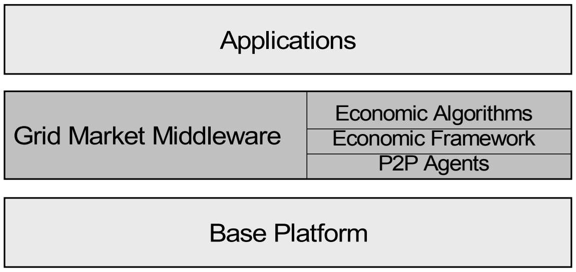

Figure 2: Layered middleware architecture

Applications interact with the *Grid Market Middleware* in order to obtain the Grid services required to fulfil the application tasks. The *Base Platform* supports the application by providing a hosting environment for the Grid services.

Within the GMM, the *Economic Algorithms* layer implements the high level economic behaviour, such as: negotiation and agent strategies used by the trading agents. The *Economics Framework* layer isolates these economic algorithms from the lower level technical details, specific to the base platform being used. The *P2P Agents* layer provides decentralized resource discovery and network topology maintenance. A detailed description of the middleware architecture can be found in [7].

The Economic Framework is modelled after the Minimal Market Model (MMM) [8]. The MMM builds a market vocabulary based on structural similarities that many market mechanisms have in common. Such a vocabulary is needed for our message specifications. Based upon this framework, diverse markets models can be supported simultaneously. These vary from direct agent to agent bargaining towards double sided auctions.

The main concept of the *Economic Framework* is that of an intention. In this context, an intention is meant as the willingness to establish an agreement on the usage of a Grid resource. Agents that join a market express their intention to sell or buy resources. Agents receive other agents' intentions and can establish an agreement. Furthermore, agents can join the market with the role of clearing the intentions of other agents. If a seller is also a clearer of buy intentions, the seller becomes an auctioneer. A pure auctioneer clears the intentions of both sellers and buyers. This allows participants to use the market model more suitable to its objectives and policies.

3.2 Integrating Grid applications with the GMM

Figure 3 depicts the interaction between the applications and the Grid Market Middleware, which is based on WS-Agreement standard [9].

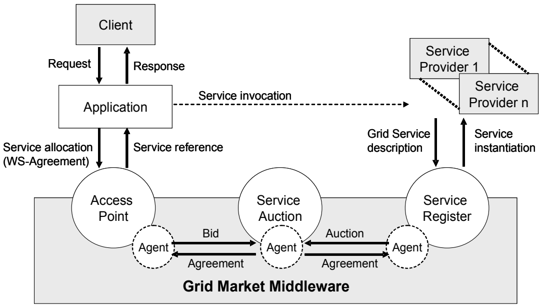

Figure 3: Interaction between the application and the middleware

From the provider side, Grid services, which represent either software services (e.g. mathematical services or data mining services) or computational resources, are registered to the middleware, resulting in the generation of an intention to sell and the instantiation of an agent to drive the corresponding negotiation.

From the application side, the requests from the user for application services are converted into series of Grid service requests. This is realized by an application specific logic. The application translates these requirements into a WS-Agreement document, which is subsequently submitted to the Grid Market Middleware. At the Grid Market Middleware, the Access Point translates this request from the application into a set of intentions to buy services and starts the agents to handle the negotiation.

The intentions from the application and service provider are matched by clearing processes which are specific of the auction model been used. Once an agreement is reached between the trading agents, a Grid service instance is created and a reference is returned to the application, which then can invoke it.

3.3 Auction based markets in the Grid Market Middleware

The following use case illustrates the integration of an auction mechanism into the middleware. As an example of auction, the MACE mechanism [6] is applied. MACE is a combinatorial exchange designed for trading Grid services and is briefly outlined in this section. Subsequently, its integration into the middleware is exemplified.

The MACE Auction

MACE allows multiple bidding agents (representing service requesters and providers) the simultaneous submission of bids on heterogonous services expressing substitutes (realized by XOR bids) and complements (realized by bundle bids). Furthermore, the mechanism is capable of handling cardinal attributes. For instance, a resource consumer can bid on a bundle consisting of a computation service and a storage service. The computing service should have two processors. Each processor should have at least 700MHz and the storage service should have at least 300 GB of free space. The bids – encoded as WS-Agreement offers – are submitted to an order book which can be traced by bidding agents. Regularly (in case of a call market) or continuously (in case of continuous matching), the auctioneer determines an allocation (winner determination) and corresponding prices.

The objective of the allocation process in MACE is the maximization of social welfare, i.e. the difference between the buyers' valuations and the sellers' reservation prices. The problem is formulated as a linear Mixed Integer Program (MIP) and, thus, can be solved by standard optimization solvers (e.g. CPLEX).

The outcome of the winner determination model is efficient allocation, as long as buyers and sellers reveal truthfully their valuations. The incentive to set bids according to the valuation is induced by an efficient pricing mechanism. MACE implements a *k*-pricing scheme and determines prices for buyers and sellers on the basis of the difference between their bids. For instance, presume a buyer wants to buy a computation service for 5 and a seller wants to sell a computation service for at least 4. The difference between these bids is $\theta=1$, where θ is the surplus of this transaction and can be distributed among the participants. This schema is applied to MACE and results in an approximate efficient outcome.

Implementation of a MACE auction in GMM

A MACE auction is created when agents enter the market as a clearer of both sell and buy intentions and the sellers and buyers policies dictate that their intentions must be forwarded to clearers. Figure 4 shows a conceptual description of the MACE protocol under the GMM in a scenario where there is one auctioneer and one direct seller that establishes it own auction:

1. The MACE auctioneer joins the market as a clearer for both, sell and buy intentions. The GMM advertises it to other participants.
2. The direct seller expresses its intention to sell and to clear this sale. It is advertised to the other participants as a clearer for buy intentions.
3. A seller, which is not a clearer, expresses its intention to sell, which is forwarded to all known clearers (in this case, only the auctioneer)
4. The buyer expresses its intention to buy, which is forwarded to all known clearers (in this case, both the auctioneer and the direct seller)
5. The direct seller receives the intention to buy and tries to match it with its intention to sell, but there is not a concordance.
6. The auctioneer receives the intention to buy and matches it with the intention to sell of the seller. An agreement is sent to both participants.

This scenario shows the basic concepts behind the middleware model: the advertisement of roles and intentions and the matching (clearance) of intentions.

The GMM knows all interchanges of intentions and resulting agreements and can prevent, for instance, a buyer to establish agreements of the same intention in more than one clearer, or the clearer to clear the same intention more than once.

3.4 Security considerations

One important consideration to allow the creation of such an open market environment is that of security. However, no single technology can address all the security requirements of an open Grid market and a blend of middleware and application protocols based on open standards is needed [10]. Furthermore, each participant's administrative domain might require different policies to be applied, e.g. different digest algorithms or different key lengths for encryption. Therefore, the security for the Grid Market Middleware must address the following main objectives:

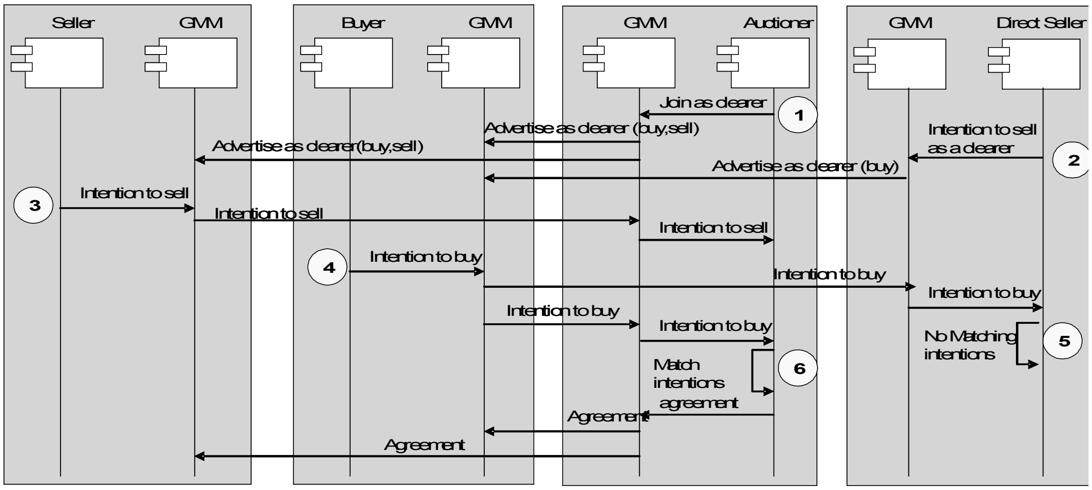

Figure 4: Auction protocol in the GMM

- Allow the adaptation of the security to each administrative domain policy and also to the requirements of the individual applications
- Maintain the security infrastructure open to emerging security standards
- Be able to inject the security into any negotiation protocol transparently and to keep the application and service providers isolated from its complexity
- Leverage the security mechanisms provided by the underlying platform.

To achieve these goals, we have devised a general security framework which is based on policies that dictates how the different security mechanisms are used on each negotiation process. These policies are enforced transparently by the Grid Market Middleware along the interaction between the application and the service provider.

4. Considerations for implementing Grid Auctions

In general, different types of applications and users in a Grid form a Virtual Organization (VO) for planning, scheduling, and coordination phases within specific projects or businesses. A Grid infrastructure allows the users of a VO to interact among them for the duration of that particular VO. The ability of market based coordination is to adjudicate and satisfy the needs of VOs, in terms of policy adherence and quality of service levels. The requirements upon such a coordination mechanism are manifold, i.e. applications and users have different requirements upon auction mechanisms and its underlying institutional rules. In this section we will introduce some of these requirements and describe how the proposed architecture can handle them.

From an abstract point of view, the Grid Market Middleware is used by an application to find suitable auction instances. These instances are required in order to establish an agreement with a counterpart which fulfils pre-specified contractual conditions, policies, and QoS levels. Once such an agreement is reached between the trading agents, a Grid service instance is created for the application and a reference is returned to the application, which can invoke that service.

To take full advantage of this market mechanism, it will be necessary to include some logic at the application level by specifying a maximum price for a Grid service type the application is interested in. This maximum price is like a budget for the buyer service agent that could be spent to buy basic services and resources. The logic at the application level will include a policy document on which the decision is taken by the application to accept or reject the Grid service instance that has been found on the market. The policy document in this case will take the “price” as the characteristic to be taken into the consideration and this is done by checking that this is less than the budget proposed initially by the application.

A typical Grid application will specify more complex resource allocation requests than presented in the sample scenario, including bundles of the type {*Budget, (CPU (1000cycles), Memory (1 GB), Disk (300Mb), Medical Simulator*} that might be satisfied either as a single allocation or as a series of partial allocations, so called co-allocations. In such scenarios, two possible strategies are considered: (i) select an auction mechanism which is capable of handling such complex application requests or (ii) select multiple single auction instances and purchase the required services one after another. The first strategy is often time advantageous, as it is ensured that all required services are allocated. An adequate auction mechanism could be the outlined MACE auction which supports bundle-bids and co-allocation constraints. However, sometimes it could be possible that not all required services are traded within an MACE auction, e.g. highly specialized services such as the required medical simulation service. In such cases, the services have to be purchased by the use of several independent auction instances. For example, one auctioneer might offer the bundle {*Price, (CPU (750 cycles), Memory (1 Gb), Disk (150 Mb)*}, while other auctioneer has another amount of resources and the medical simulator service. A more complex policy document at the application level will be invoked as a decision has to be taken by the application to describe the acceptable allocations: for instance, if one resource can be split or not and the minimum meaningful allocation. This co-allocation must be coordinated by the middleware even when each service is allocated by one independent auction. This coordination is still an open issue subject of future work.

5. Related Work

The use of market mechanisms for allocating computer resources is not a new approach. Several different auction mechanisms and negotiation strategies for distributed computing infrastructures and multi agent systems have been proposed in literature [e.g., 3, 4, 5, 6, 11, and 16]. However, only little work has been proposed for integrating the markets into a technical middleware.

Popcorn was one of the first systems providing a Web based user interface for trading computer resources via different auction mechanisms [4]. Users can deploy their applications using the Popcorn framework. They can bid on several computing machines in order to execute their applications. Popcorn includes a repeated Vickrey auction, a double auction and a clearing house double auction. However, Popcorn is a proprietary system and can thus not be integrated into current state-of-the art Grid middleware.

OCEAN (Open Computation Exchange and Network) [11] provides an open and portable software infrastructure for automated commercial buying and selling of computing resources over the Internet. Each OCEAN node that wants to buy resources uses a matching service which implements an optimized peer-to-peer search protocol. This protocol is used to find a set of potential sellers based on the description of the resources being requested. Afterwards, an automatic negotiation process starts with each seller, based on the rules dynamically defined in a XML format. The ability to define negotiation rules is a remarkable characteristic of OCEAN that allows the adaptation of the economic model to diverse applications. The main limitation we found is that the infrastructure does not consider auction nodes that offer this service to other nodes that do not want (or can not) establish it for themselves.

Tycoon [12] is a distributed market-based allocation architecture based on a local auctioning for resources on each node. Auctioneers receive fine grained requests of local resources from agents acting on behalf of applications and schedule them using efficient sealed bid auctions. This is realized in a way that approximates a proportional share algorithm, allowing high resource utilization rates and the adaptation to changes in demand and/or supply. One interesting feature of Tycoon is that it separates the allocation mechanism from the agents which interprets application and user preferences. Thus allows the specialization of agent to different applications. Tycoon, however, does not offer any framework for the construction of those agents. A major limitation of Tycoon is that the resource allocation mechanism is already fixed in the system design and no extension or adaptation methods are offered. To overcome this limitation, our proposed framework is capable to plug key components to adapt to specific application domain in environments with heterogeneous or changing resource allocation requirements. Furthermore, we offer a set of high level tools to develop those components, alleviating the implementation burden for new market designs.

6. Conclusion

The paper at hand proposes a flexible architecture for integrating auctions into Grid environments. In contrast to other approaches, the proposed architecture accounts for the variety of different auction types and the need for several simultaneous instances of auction mechanisms. The architecture hides the complexity of underlying Grid middleware toolkits and thus simplifies the instantiation of auctions for the Grid.

The requirements upon the architecture are elicited by means of a scenario. Subsequently, the basic concepts of the architecture are outlined. The architecture and the corresponding protocols are illustrated by an example. The importance of the architecture is further discussed by some application considerations. The architecture is currently being implemented within the CATNETS[3] project.

Future work has to include the integration of several auction instances as a proof-of-concept. Furthermore, we have to analyze technical and economic issues resulting from running several concurrent auction instances in parallel. This may lead to a fragmentation of liquidity and, as such, a low success rate. In such cases, Grid specific requirements such as co-allocations have to be taken into account.

The existence of several different markets may lead to a splitting of liquidity between all markets. Thus, the consequences of this liquidity loss have to be analyzed, e.g. as proposed in [13]. The architecture will be combined with current agent technologies in order to automate the purchasing process, e.g. using the work presented in [14].

---

[3] See http://www.catnets.org/ for details.

Acknowledgements

This work has partially been funded by EU IST programme under grant 003769 CATNETS.

References

[1] Foster, I., “What is the Grid? A Three Point Checklist”, Working Paper, 2002
[2] Foster, I., Kesselman, C., Nick J.M, Tuecke, S. „Grid Services for Distributed System Integration”, IEEE Computer 35 (6), 2002, pp. 37-46
[3] Buyya, R., Abramson, D., Venugopal S., “The Grid Economy”, Proceedings of the IEEE 93 (3), 2005, pp. 698-714
[4] Nisan, N., London, S., Regev, O., Camiel N., “Globally Distributed Computation over the Internet – The POPCORN Project”, Proceedings of the 18th International Conference on Distributed Computing Systems, 26 - 29 May, 1998
[5] Joita, L., Rana, O.F., Gray, W.A., ”A Double Auction Economic Model for Grid Services”, Proceedings of Euro-Par 2004 Parallel Processing - 10th International Euro-Par Conference, Pisa, 2004
[6] Schnizler, B., Neumann, D., Veit, D., Weinhardt, C., “A Multiattribute Combinatorial Exchange for Trading Grid Resources”, Proceedings of the 12th Research Symposium on Emerging Electronic Markets, Amsterdam, 2005
[7] Ardaiz, O., Chacin. P, Chao, I., Freitag, F., Navarro. L., “An Architecture for Incorporating Decentralized Economic Models in Application Layer Networks”, International Workshop in Smart Grid Technologies, Utrecht, Netherlands, July 25 - 29, 2005.
[8] Rolli, D., Neumann, D., Weinhardt, C., “A Minimal Market Model in Ephemeral Markets”, Applying Formal Methods: Testing, Performance and M/ECommerce, FORTE 2004 Workshops, LNCS 3236: pp. 86-100, 2004,
[9] Web Services Agreement Specification (WS-Agreement), 28 June 2005, GRAAP Working Group at the Grid Forum. https://forge.gridforum.org/docman2/ViewCategory.php?group_id=71&category_id=659
[10] Naedele, M., “Standards for XML and Web Service Security”, IEEE Computer, April 2003
[11] Padala, P., Harrison, C., Pelfort, N., Jansen, E., Frank, M. P., Chokkareddy, C., “OCEAN: The Open Computation Exchange and Arbitration Network, A Market Approach to Meta computing.” Proceedings of the International Symposium on Parallel and Distributed Computing, 2003
[12] Lai, K., Huberman, B.A., Fine, L., “Tycoon: A Distributed Market-based Re-source Allocation System,” HP Lab, Palo Alto, Technical Report cs.DC/0404013, Apr. 2004.
[13] Czernohous, C., Kolitz, K., Mäkiö, J., Weber, I. Weinhardt, C., “Integrating Electronic Market Models - Problems and Solutions of Parallel Markets”, 10th Research Symposium on Emerging Electronic Markets, 2003
[14] Preist, C., Byde, A., Bartolini, C., “Economic dynamics of agents in multiple auctions”**,** Proceedings of the fifth international conference on Autonomous agents, pp. 545-551, 2001
[15] Shetty, S., Padala, P., Frank, M. “A Survey of Market-Based Approaches to Distributed Computing”. Technical Report. *Computer & Information Science & Engineering, University of Florida,* 2003.
[16] Sandholm, T. “Distributed Rational Decision Making” in Weiß, G. (ed) “Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence, pp. 201-258, 1999