Market Formulation for Resources Allocation in an Ad-hoc Grid

Behnaz Pourebrahimi, Luc Onana Alima*, Koen Bertels
Computer Engineering Laboratory, Delft University of Technology, The Netherlands
{b.pourebrahimi, k.l.m.bertels}@tudelft.nl

Service des Systémes Distribués, Institut d'Informatique, Université de Mons-Hainaut, Belgium*
luc.onana@umh.ac.be

Abstract

An ad hoc grid is a spontaneous formation of cooperating heterogenous computing nodes which attempts to provide computing resources on demand to every participant. In this paper, we study market formulation for resource allocation in an ad-hoc Grid. Continuous Double Auction (CDA) protocol with discriminatory pricing policy is selected as the market protocol and a novel bidding mechanism is presented to determine ask/bid prices. We study the performance of our bidding mechanism and compare it with three other mechanisms from the literature. The performances are investigated in the terms of price stability, throughput, and load balancing. The experimental results show that our mechanism outperforms other bidding mechanisms in similar conditions.

1 Introduction

Ad-hoc Grid is a type of Grid with the aim of harnessing unused computational resources inside or across organizations. In an ad-hoc Grid, every node in the network can spontaneously arise as a resource consumer or a resource producer at any time when it needs a resource or it possesses an idle resource. In this paper, we study market formulation for resource allocation in this type of Grids. In a market-based Grid, resources are treated as the commodities. The individuals participating in the Grid for trading resources, are potential buyers or sellers. Market formulation for Grid resource allocation is the processes of determining trade prices (offer, bid and transaction prices) and defining a market protocol for matchmaking between consumers and producers of resources.

For resource allocation in an ad-hoc Grid, a market protocol has to be selected to allow many resource owners (sellers) and resource consumers (buyers) to trade simultaneously. For this purpose, we choose the Continuous Double Auction (CDA) protocol. In a market formulation, besides the market protocol, the bidding mechanism for consumers and producers of resources has to be defined. In this paper, we present a novel bidding mechanism for resource consumers and producers to set the bid and ask prices. We compare the performance of our mechanism with three existing bidding mechanisms in the literature, namely, Zero Intelligence (ZI) [6], Zero Intelligence Plus (ZIP) [4] and Gjerstad and Dickhaut (GD) [5]. The experimental results show, our mechanism is outperforming the other mechanisms considering the performance criteria which are important to the users and the system.

The paper is structured as follows. Section 2 gives an overview of different market models and specifically discusses double auction protocols. In Section 3, we present three bidding mechanisms from the literature and introduce our bidding mechanism. System architecture is discussed in Section 4. In Section 5, we perform experimental evaluations and compare our bidding mechanism against the three other mechanisms. We conclude the paper in Section 6.

2 Market Protocols

For resource allocation in ad-hoc Grid, we need a protocol that supports simultaneous participation of producer/consumer, observes resource/request deadlines and can accommodate the variations in resource availability. Two main classes of market models used for Grid resource allocation are commodity markets and auction models [10][7]. In commodity markets, resource prices are decided from the overall demand and supply in the system. A central planner gathers all the supply and demand information to generate a set of equilibrium prices. This type of market attempts to satisfy all consumers and producers in a given price (equilibrium price) within a time frame. Commodity markets are not suitable model for resource allocation in ad-hoc Grids where the resources are not dedicated and the supply and demand in the system is dynamic. As

in such networks, the complexity of implementing a centralized market which rely on the aggregate supply and demand becomes very high. Competitive and decentralized nature of auction models makes them an appropriate choice for resource allocation in ad-hoc Grids.

Considering different auction protocols, English auction and Dutch auction are sequential and are based on open-cry where each bid has to be broadcasted to all participants. This becomes a considerable communication overhead in the context of ad-hoc Grids. Moreover their inability to observe time deadlines and no support for the simultaneous participation of producer/consumer are the reasons that make them unsuitable for resource allocation in ad hoc Grids. First-price auction and Vickrey auction are simultaneous and close bid auctions, but they are one-to many auctions. To support simultaneous participation of both consumers and producers in an ad-hoc Grid, a many to many protocol is needed. Double auction is a two sided auction in which buyers and sellers are treated symmetrically with buyers submitting requests and sellers submitting offers.

2.1 Continuous Double Auction Protocol

There are two forms of double-sided auctions. Whether the auction is continuous-time or discrete-time makes the most difference when distinguishing between them. In a **discrete-time auction** which also is named periodic double auction, all trades move in a single step (a predefined time frame) from initial allocation to final allocation. The Clearing house (CH) or Call auction is a representative example of a discrete-time double-sided auction. In CH auction, consumers and producers submit their sealed bids and asks and these are integrated into revealed supply and demand curves, from which a market-clearing price is determined. This pricing policy is called **uniform-price policy** in which all exchanges occur.

In contrast with discrete-time auctions, a **continuous-time auction** permits exchanges at any time and overall net trade is typically composed of many bilateral transactions at different prices. The primary example of a continuous-time double-sided auction is the Continuous Double Auction (CDA). There is no clearing-time frame in a continuous-time double auction. Bids and asks are continuously received and matched and trades can occur at any time. In this scheme, a participant may make a bid or ask at any time but, once made, it persist until it is accepted or the trader chooses to alter it. This type of auctions uses a **discriminatory policy** in which the prices are set individually for each matched buyer-seller pair. For resource allocation in an ad-hoc Grid, we select a continuous double auction protocol with discriminatory policy to fulfill the deadline constraints for requests and offers more efficiently. In this model, the transaction prices are computed based on the bid/ask prices.

3 Bidding Mechanisms

Consumer and producer agents adopt a bidding mechanism to define their bid and ask prices. In the literature, we can find several bidding algorithms for consumer and producer agents in Double Auction markets [5, 9, 3]. In this section, first we briefly study three most known bidding mechanisms and then we present our bidding mechanism.

3.1 Zero Intelligence (ZI)

In the ZI approach [6], agents use a simple strategy. In this strategy, agents generate random order prices, ignoring the state of the market. In this paper, we consider a ZI-C (Zero Intelligence-Constraint) model in which traders are subject to a budget constraint and consumer bids and producer asks are limited between a maximum and a minimum price. ZI-C sellers can only make offers in the range between their limit price and a maximum price, and ZI-C buyers can only make bids in the range from a minimum up to their limit price.

3.2 Zero Intelligence Plus (ZIP)

In the ZIP strategy [4], each agent maintains a scalar variable denoting its desired profit margin, and it combines this with a unit's limit price to compute a bid or ask price. At the start of trading, the price is initialized to a random positive surplus value (a private value), and it is adjusted after every successful or failed trade. For the successful trades, price is adjusted towards the trade price. For failed bids, buyers adjust price in the direction of beating the failed bid. Sellers behave similarly for failed asks [1].

3.3 Gjerstad and Dickhaut (GD)

The Gjerstad-Dickahaut (GD) is a memory based trader algorithm for CDA, described in [5] and refined in [9]. GD traders have a strategy for shout price selection based on maximizing expected profit. The maximization of expected profit relies on the GD trader forming a belief and payoff function. GD agents [5] use the history H_M of recent market activity (the bids and asks leading to the last M trades) to calculate a belief function $f(p)$ estimating the probability for a bid or ask at price p to be accepted.

3.4 A Learning and Adaptive Mechanism for BiDding Agents (λ)

We propose a learning and adaptive bidding mechanism in which consumer and producer agents determine their bid and ask prices. This mechanism is denoted as λ in this paper. The pricing mechanism presented defines a logical

price by local analysis of the previous trade cases. The price is defined as the price of each unit of resource that consumer and producer agents are willing to buy or sell. Let denote by $p_b(t)$ the bid price of a consumer agent at time t and $p_a(t)$ the ask price for a producer agent at time t. We assume that each consumer agent has a maximum bid price, denoted max_b, for a resource unit. This maximum price is determined by the node's budget. On the other side, each producer agent has a minimum ask price, denoted min_a, below which the node is not willing to sell a unit of resource. More formally, we have that:

$$\forall t, \quad \begin{array}{l} p_a(t) \geq min_a \text{ and} \\ p_b(t) \leq max_b \end{array} \tag{1}$$

In our bidding algorithm, agents update their ask (respectively bid) prices using the experience they gained from their previous utilization of the Grid. Informally, the idea is as follows. If an agent has not been successful in buying resources, at current time the agent updates its bid price in a way that tends to increase its chance to buy resources in the future. If an agent has been successful, it conservatively continues to bid in a way that ensures its chance of buying resources in the future. A seller agent behaves in a similar manner. If it has not been successful in selling resources, at current time it updates its ask price in a way that increases its chance of selling its resources in the future. Otherwise, it behaves in a conservative manner.

Formally, the ask price of a producer agent at time t is computed according to the assignment in (2), while the bid price for a consumer agent is given by assignment (3).

$$p_a(t) \leftarrow \max\{min_a, p_a(t-1) + \alpha.p_a(t-1)\} \tag{2}$$

$$p_b(t) \leftarrow \min\{max_b, p_b(t-1) + \beta.p_b(t-1)\} \tag{3}$$

where α and β are coefficients which determine the rate at which the price is increasing or decreasing. We name these parameters *reinforcement parameters*, as they reinforce the learning and apply two aggressive and conservative bidding strategies. These parameters are set according to variations in task or resource utilization at each individual node.

For a given node, we define the *task utilization* as the ratio of allocated tasks to all submitted requests and the *resource utilization* as the ratio of allocated resources to all submitted offers. Formally, let $T = [s, e]$ be a time period. We shall call s the start of T and e will be called the end of T. Let *ru*(T) and *tu*(T) be the resource and task utilization, respectively, over the time period T.

For a given time period T, the resource utilization is formally defined by Equation (4), while the task utilization is formally given by Equation (5).

$$\textit{ru}(T) = \frac{S(T)}{N_o(T)} \tag{4}$$

$$\textit{tu}(T) = \frac{P(T)}{N_r(T)} \tag{5}$$

where $S(T)$ and $P(T)$ are respectively the total numbers of sold and purchased resources in the time period T. $N_o(T)$ and $N_r(T)$ are respectively the total numbers of offered and requested resources in the time period T.

We now define variations in resource and task utilization. To this end, let $T_1 = [s_1, e_1]$ and $T_2 = [s_2, e_2]$ be two consecutive time periods such that $e_1 = s_2$ and e_2 is the current time. We capture variations in resource (respectively task) utilization from period T_1 to T_2 by the following equations.

$$\Delta ru_{(T_1 \rightarrow T_2)} = ru(T_2) - ru(T_1) \tag{6}$$

$$\Delta tu_{(T_1 \rightarrow T_2)} = tu(T_2) - tu(T_1) \tag{7}$$

We now define the reinforcement parameters as follows:

$$\alpha = \begin{cases} -(K - (ru(T_2))^2)^2 & if\ \Delta ru_{(T_1 \rightarrow T_2)} \leq 0 \\ L * (ru(T_2))^2 & if\ \Delta ru_{(T_1 \rightarrow T_2)} > 0 \end{cases} \tag{8}$$

$$\beta = \begin{cases} (K - (tu(T_2))^2)^2 & if\ \Delta tu_{(T_1 \rightarrow T_2)} \leq 0 \\ -L * (tu(T_2))^2 & if\ \Delta tu_{(T_1 \rightarrow T_2)} > 0 \end{cases} \tag{9}$$

where K and L respectively define the maximum rate of aggressive and conservative bidding. Aggressive bidding is defined as a strategy using which agents increase or decrease their prices in a high rate. On the other side, a bidding is conservative when agents change their prices in a low rate. The aggressive and conservative bidding strategies are taken in the networks where there is not a balance between the supply and the demand. In Our experiments, we have considered $K = 1$ and $L = 0.1$. The values of K and L can be changed individually by nodes based on the user preferences such as budget consumption, urgent tasks and etc. For instance, if a node has very limited budget, it should choose lower level for aggressive bidding and avoid very high bid prices. On the other hand, in case of very urgent tasks, nodes can always increase level of their aggressive bidding or decrease level of their conservative bidding. The functionality of these parameters have been shown in our previous work [8].

4 System Architecture

Our model is composed of three types of agents (see figure 1): *Consumer (buyer)*, *Producer (seller)* and *Auctioneer*. There is one consumer/producer agent per node. A consumer/producer agent controls the process of buying/selling resources by estimating the required resources for executing the tasks or availability of the resources, calculating the price and generating and submitting a request/offer for the corresponding task/resource.

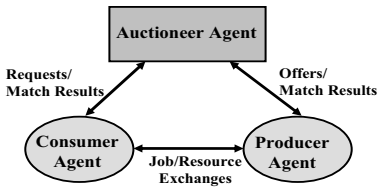

Figure 1. System Components.

The auctioneer agent controls the market using a Continuous Double Auction protocol. The market works in the following simple manner: the buyers and sellers announce their desire to buy or sell resources to the market. Auctioneer continuously receives the requests and offers and keeps them in its depositories. An auctioneer finds the matches between buyers and sellers by matching offers (starting with lowest price and moving up) with requests (starting with highest price and moving down). When a task query arrives at the market place, the protocol searches all available resource offers and returns the best match which satisfies the task's constraints. These include resource quantity, time frame and price. If no match is found, the task query object is stored in a queue. The queries are kept in the queue till the time to live (TTL) for them is expired or a match is found. When a resource becomes available and several requests are waiting, the one with the highest price bid is processed first. A discriminatory pricing policy is used to calculate transaction prices. In this policy, there is no global equilibrium price and the transaction prices are set individually for each matched buyer-seller pair as the average of bid and ask prices.

4.1 Request/Offer Message Specification

Each request/offer submitted by consumer/producer agents contains the consumer/producer identifications, resource requirements and attributes, and time and cost constraints. A request message contains:

- Consumer Id: identifies the consumer node (e.g. IP address).
- Request-Id: a unique number for each request to distinguish between different requests from a consumer.
- Resource-type: the type of resource that is required, such as CPU, Memory, Storage.
- Resource-quantity: the quantity of required resource.
- Request-TTL: the time to live for the request.
- Bid-price: the maximum price that consumer is willing to pay for each unit of resource.
- Consumer-Budget: the current budget of consumer.

An offer message includes these information:

- Producer Id: identifies the producer node (e.g. IP address).
- Offer-Id: a unique number for each sent offer to distinguish between different offers from one producer.
- Resource-type: the type of resource that is offered, such as CPU, Memory, Storage.
- Resource-quantity: the quantity of offered resource.
- Offer-TTL: the time during which the offer is valid.
- Ask-price: the minimum price that consumer is willing to receive for each unit of resource.

4.2 Response Messages

Whenever the auctioneer finds a match between an offer and a request message, it calculates the transaction price and sends back two response messages to the corresponding consumer and producer. The response message from auctioneer to consumer has these specifications:

- Producer Id: discovers the producer from which to buy the required resource.
- Request-Id: serves to match this response with the corresponding request that was previously submitted.
- Offer-TTL: is the remaining time during which the offered resource is available.
- transaction-price: is the price that consumer has to pay to producer for each unit of resource.

A response message to producer contains:

- Consumer Id: is used by receiving producer to establish a contact with a consumer to sell its resource.
- Offer-Id: serves to establish the correspondence between this offer and the matching request.
- Request-TTL: is the remaining time by which the requested task has to be completed.
- transaction-price: is the price that producer receives from consumer for each unit of resource.

5 Performance Evaluation

To evaluate the performance of different bidding algorithms, we set up a Grid like environment in a local network. Our application test-bed is developed using J2EE and Enterprise Java Beans. Auctioneer is deployed on JBoss application server.

	Average Trans. Price	StDev Transaction Price	Throughput	StDev Task Util.	Load Balancing
λ	14.95	6.75	%92	3.6	%96
ZI	47.26	34.20	%80	5.4	%93
ZIP	25.80	10.24	%61	33.4	%45
GD	29.95	1.06	%50	37.9	%24

Table 1. Comparing the four Mechanisms.

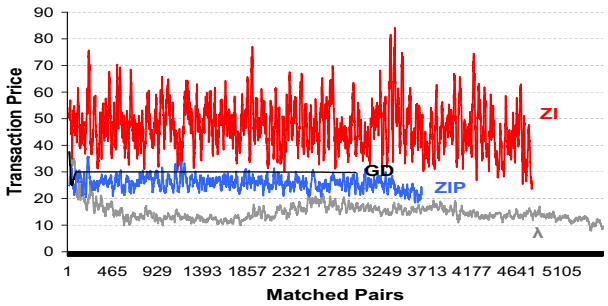

Figure 2. Transaction prices .

5.1 Experimental Setup

In our simulation platform, a node sends a request/offer through the consumer/producer agent when it needs some computational resource for running its tasks or it has some computational resource available. We assume that the tasks are atomic and can not be divided, therefore each request is matched with only one offer.

The experiments are performed in a local ad-hoc Grid with 60 nodes. So, there are 60 consumer agents and 60 producer agents participating in the market. All nodes are assigned equal budgets when joining the grid. The limited budget defined for each node can be used to trade for required resources. It's also possible to earn credits by devoting the idle computational resources for demanding consumers. Each node creates 100 requests and 100 offers during the simulation in a random order. The CPU time is considered as the resource. For a request, resource requirement is expressed in the term of CPU time for execution of the task and TTL determines the time during which the task has to be executed. For an offer TTL is the time during which the CPU is available. These values are generated randomly for each request/offer in a specific range.

5.2 Comparing Bidding Algorithms

In this section, we study performances of the four bidding mechanisms in a CDA market. Experiments are performed in the completely similar condition for each of the four approaches. The system and user centric metrics considered as the basis for performance evaluation are: price stability, system throughput, and load balancing.

5.2.1 Price Stability

Price stability is an important parameter in markets. Stable and lower prices are always intendancy of every efficient market. To study the price stability in the market, we present the variation in transaction prices by measuring standard deviation of transaction prices in the network.

Standard deviation of transaction prices for each approach is presented in table 1. The results show that GD approach has highest price stability among the four. GD traders record all asks and bids made in the history of the last M transactions. So, in this approach an equilibrium is reached after few rounds. λ approach and ZIP also show an acceptable price stability while ZI presents a high variation in prices which is the consequence of random generation of bids and asks. To study the values of transaction prices, we measure the average transaction price in each approach. With comparing these values in table 1, we observe that λ mechanism gives the lowest transaction prices while ZI presents the highest among the four approaches.

Figure 2 depicts transaction prices evolution during the simulation time in each approach. From the figure, we can see that ZI approach gives the highest price fluctuation and also the highest price values, GD shows the most stable prices and λ approach gives the lowest price values with low fluctuation in prices. The differences in the lengths of graphs is due to differences in the number of found matches in the four approaches.

5.2.2 Throughput

The throughput of the system is measured in the terms of task/resource utilization. **Task Utilization** is defined as the percentage of tasks that are allocated to available resources. **Resource Utilization** is defined as the percentage of available resources that are used by the allocated tasks.

We measure the overall task/resource utilization in the system for each approach. As the total number of generated tasks and resources are equal, the overall task utilization in the system is equal to resource utilization. The throughput presented in the table 1 detects that λ mechanism gives highest throughput compared to other mechanisms. The simple and non intelligence approach ZI provides more throughput compared to ZIP and GD approaches.

5.2.3 Load Balancing

Load balancing attempts to distribute the computation load across multiple nodes as evenly as possible with the objective to improve performance. To study the tasks distribution

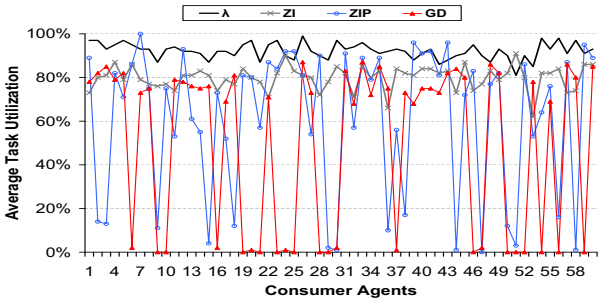

Figure 3. Individual's task utilization.

among the resources, we measure the average task utilization at individual consumer agents. Figure 3 shows average task utilization at each of 60 consumer agents in the experiment. From the figure, we can see that the task utilization is almost equally distributed among different agents in λ approach. Among three other approaches, ZI approach shows more balanced task utilization than two ZIP and GD approaches. In ZIP, we see some agents receive a high and some others very low degree of task utilization. The variation in task utilization is also high in GD, such as some agents have got zero task utilization. The values of these variations are shown in table 1 by measuring the standard deviation of task utilization.

Using standard deviation, we calculate the Relative Standard deviation (RSD) as a measurement for load balancing at the system [2]. RSD is calculated as below:

$$RSD = (stdev/mean) * 100\% \tag{10}$$

where $mean$ is the average task utilization of all nodes in the system and $stdev$ is standard deviation of the task utilization. Now, we define the level of load balancing of the system using RSD as follows:

$$\eta = (100\% - RSD) \tag{11}$$

The most effective load balancing is achieved when $\eta = 100\%$ which is obtained when $stdev = 0$. As results in the table 1 show, λ mechanism presents the lowest standard deviation and in consequence the highest load balancing. Among the other three approaches, ZI shows better performance than ZIP and GD regarding load balancing. ZIP and GD present a low level of load balancing. It is concluded that our approach provides a fair access to the resources for all nodes in the network compared to other approaches.

6 Conclusion

In this paper, we presented a market formulation for resource allocation in ad-hoc Grids where the availability of tasks and resources are dynamic. We discussed different market models and bidding mechanisms that can be employed in a market formulation for ad-hoc Grids. The market should support the simultaneous participation of resource consumers and resource producers while observing the deadline constraints. To fulfill these requirements, we adopt a Continuous Double Auction (CDA) protocol as the market protocol. We presented a novel bidding mechanism to determine the ask and bid prices. We performed experiments to compare the performance of our bidding mechanism (λ) against that of the Zero Intelligence (ZI), Zero Intelligence Plus (ZIP) and Gjerstad and Dickhaut (GD). The experimental results show that our bidding mechanism presents a good performance in an ad-hoc Grid compared to other strategies. Our approach provides low and stable transaction prices, high throughput, and a fair access to resources for all agents in the network.

References

[1] A. J. Bagnall and I. E. Toft. Zero intelligence plus and gjerstad-dickhaut agents for sealed bid auctions. In *Workshop on Trading Agent Design and Analysis*, pages 59–64, 2004.

[2] J. Cao, D. P. Spooner, S. A. Jarvis, and G. R. Nudd. Grid load balancing using intelligent agents. *Future Gener. Comput. Syst.*, 21(1):135–149, 2005.

[3] B. J. Cliff, D. Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets. Technical Report 97-141, HP, 1997.

[4] D. Cliff. Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical Report 97-91, HP, 1997.

[5] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22:1–29, 1998.

[6] D. Gode and S. Sunder. Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality. *The Journal of Political Economy*, 101(1):119–137, 1993.

[7] U. Kant and D. Grosu. Double auction protocols for resource allocation in grids. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, pages 366–371, 2005.

[8] B. Pourebrahimi and K. Bertels. Adaptation to dynamic resource availability in ad hoc grids through a learning mechanism. In *CSE '08: Proceedings of the 2008 11th IEEE International Conference on Computational Science and Engineering*, pages 171–178, 2008.

[9] G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 206–209, New York, NY, USA, 2001. ACM.

[10] R. Wolski, J. Brevik, J. S. Plank, and T. Bryan. Grid resource allocation and control using computational economies. In F. Berman, G. Fox, and A. Hey, editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 2003.