A Multi-lane Double Auction for Economic-Based Service Management in the Cloud

Xavier Vilajosana, Daniel Lázaro, Angel. A. Juan, and Joan Manuel Marquès

Abstract. Economic models have shown their suitability to allocate resources efficiently, considering an unbalanced supply and demand. As the use of the Cloud is extending, a numerous set of distributed resource allocation frameworks have been developed to attain efficient resource management while keeping the scalability of the infrastructure. However, those frameworks make use of either simple double auction mechanisms or complex approximations to the NP-complete problem of the combinatorial auction. The problem of those mechanisms is that of its generality, that is, they have not been specially designed for the trading of time-leased computational resources. In this paper we present a novel variant of the double auction that has been specially adapted to trade time-differentiated items as Cloud services can be considered. The paper presents the data structures, algorithms and architecture of the economic mechanism as well as it presents the evaluation of the mechanism through simulation. Simulated results are compared with the main double auction implementations found in the literature. The paper constitutes an approach to improve efficiency of service management and allocation in the Cloud from the point of view of the economic model and not from architectural aspects addressed by most of the contributions found in the literature.

1 Introduction

Auction mechanisms have been used to allocate resources in computational environments [Lai et al(2005)Lai, Rasmusson, Adar, Zhang, and Huberman, Eymann et al(2003)Eymann, Reinicke, Ardaiz, Artigas, Freitag, and Navarro, Buyya and Venugopal(2004), Neumann et al(2007)Neumann, Stößer, Anandasivam, and Borissov, Consortium(2008), Haussheer and Stiller(2005)]. Most of the used

Xavier Vilajosana · Daniel Lázaro · Angel. A. Juan · Joan Manuel Marquès
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
e-mail: {xvilajosana,dlazaroi,ajuanp,jmarquesp}@uoc.edu

© Springer-Verlag Berlin Heidelberg 2010

economic institutions are the Double Auction (DA) or the Combinatorial Auction (CA). The DA is a simple but powerful mechanism to allocate a single type of item amongst multiple buyers and multiple sellers. Combinatorial Auctions [Liu and He(2007), Radhanikanth and Narahari(2009), Schnizler and Neumann(2007)], in contrast, are NP-complete mechanisms that can deal with bundled items and substitute preferences. Both mechanism have been used with success in the allocation of resources in the Grid, however neither CAs nor DAs have been improved to fit to the nature of the services in the Cloud (heterogeneous and strictly time-leased). As already stated, DAs can only trade a single type of resource per instance, even multiple units, this constitutes a limitation for the allocation of services, mainly heterogeneous and differentiated by time. On the other hand, Combinatorial Auctions are not able to allocate high quantities of items due to their computational costs. Our work presents a novel approach of the Double Auction that improves the usability of Double Auctions as mechanism to allocate time-differentiated resources and services. Our approach, the Multi-Lane Double Auction (MLDA) provides the means by which multiple time differentiated items can be allocated by a single instance of the mechanisms. Besides it enables the allocation of substitute items approximating its results to the results obtained by CAs. The MLDA can be set in between both existing approaches constituting a useful alternative. It fills the gap between DAs and CAs and constitutes a good candidate when dealing with the allocation of heterogeneous computational resources/services. The paper presents the data structures, algorithms and architecture of the economic mechanism as well as it presents the evaluation of the mechanism through simulation.

2 The Double Auction

Double auctions determine the winners set amongst multiple buyers and sellers when offering to buy or sell single items. To clear the auction, the M th and (M + 1)st prices are computed, where M is the number of sell bids. It is assumed that a total order can be imposed on all the bids. This is commonly accomplished using price as the principal priority measure, and using bid quantity or bid placement time to break ties. Conceptually, finding the M th and (M + 1)st bids is simply a matter of sorting the bids in descending order, and identifying the Mth and (M + 1)st items in the list. The prices between the M th and (M + 1)st bids (inclusively) represent the range of prices for which supply balances demand. At prices in the range, the number of buyers willing to buy at that price equals the number of sellers willing to sell, with the caveat that when M th= (M + 1)st, one side or the other may have some participants who lose on tie-breaking criteria. It is important to note that this process of identifying the equilibrium price range works regardless of the relative position of the buyers and sellers in the list. The k-Double Auction computes a clearing price that is a ratio of the two boundary prices. Furthermore, the M th and (M + 1)st prices delineate the set of currently winning bids, referred to as the transaction set. Again, modulo ties at the boundaries, buyers at or above the M th-price would purchase an item if the auction cleared, and sellers at or below the (M + 1)st price would sell an item. It follows that the M th-price and (M + 1)st-price constitute exactly the information that is typically provided to participants in the form of price quotes. The M th-price is the ask quote and informs a potential buyer of the minimum that she would have to offer to be certain to enter the current transaction set. Symmetrically, the bid quote, equal to the (M + 1)st-price, informs a potential seller the maximum that he would be able to offer to become a current winner. Figure 1 exemplifies the above described procedure.

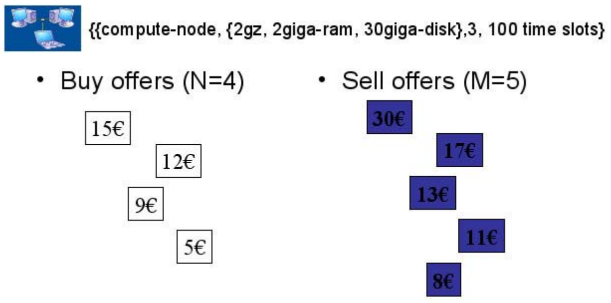
3 MLDA Functionalities

The Multi-Lane Double Auction (MLDA) has to support multiple time-differentiated items per instance while keeping the general functionalities offered by an auction. Since multiple time-differentited items must be supported, the structure of the MLDA has been designed as a set of lanes where each lane represents an item. The main operations offered by the MLDA are:

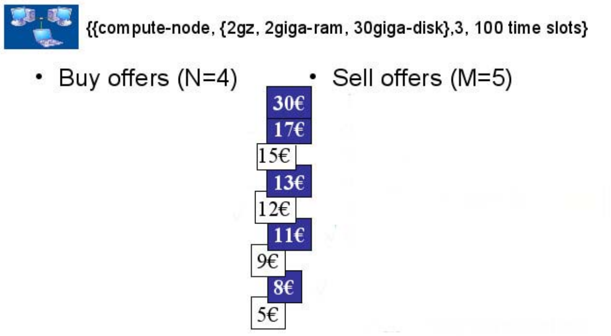
- **Insert/Remove Bid.** When a new bid is received and the auction system verifies that it satisfies whatever bidding rules exist. It must be inserted into the auctions data structures for its corresponding lane. Similarly, when a bid is withdrawn, it must be removed from the data structures. Inserted bids can be precise which means that they can only be inserted in a specific lane. Imprecise bids are those that can be placed in more than one lane.
- **Compute Quote.** The auction will generate price quote information for a lane .
- **Clear.** At designated times, the auction will compute exchanges between the buyers and sellers, notify the participants, and remove the winning bids from the data structures.

4 MLDA Operations

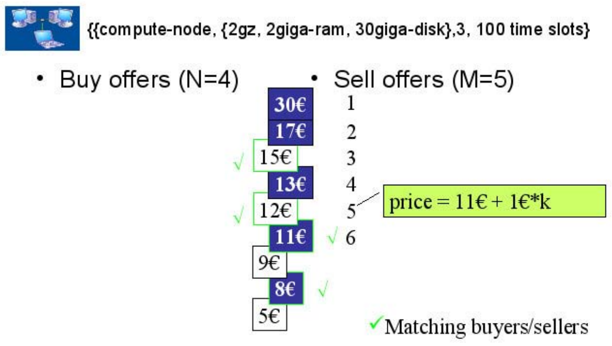
The MLDA algorithm has been developed following the same idea as the general double auction mechanism presented in the previous section. Bids and asks are organised in lanes, and for each lane the general double auction data structures are maintained. For each lane the MLDA keeps four structures to maintain bids sorted and another one global structure to buffer bids representing substitute preferences (henceforth *imprecise bids*). In the literature many different data structures can be found for such an aim: Heaps, Internal Path Trees, AVL or sorted lists. However, for the description of the MLDA algorithm it will be considered the use of sorted lists as they keep concepts simple and it does not affect the general functional behaviour of the algorithm. As indicated, each lane is represented by four sorted lists to store winning bids (Bin) , winning asks (Sin), losing bids (Bout) and losing asks (Sout). Bin is sorted in ascending order so that the lowest winning bid is the head of the list while Bout is sorted in descending order to keep the highest losing bid at the head of the list. Sin is sorted in descending order, keeping the highest winning ask at the head and Sout is sorted in an ascending order to keep the lowest losing bid at its


{{compute-node, {2gz, 2giga-ram, 30giga-disk},3, 100 time slots}
Buy offers (N=4)
Sell offers (M=5)
15€
12€
9€
5€
30€
17€
13€
11€
8€

(a) Step 1. Sort the list of buy bids and sell bids(asks)


{{compute-node, {2gz, 2giga-ram, 30giga-disk},3, 100 time slots}
Buy offers (N=4)
Sell offers (M=5)
30€
17€
15€
13€
12€
11€
9€
8€
5€

(b) Step 2. Merge both lists



(c) Step 3. Find the price. It is the Mth bid in descending order. Mth represents the number of sell bids received.

Fig. 1 Steps to compute winners in a double auction

head. By construction the MLDA keeps the Social Welfare maximum. This is the invariant of the insertion algorithm and is a key point for the economical efficiency of the MLDA. Social Welfare is a measure of wellness that permits to weight up how resources are distributed taking into account their need.

The algorithm does not differentiate between precise or imprecise bids, as a bid can be placed in one, two, three or more lanes, so the choice to where the bid will be placed will be guided by the possible lanes where the bid can be inserted. One important detail to consider and a fundamental issue to be solved by our algorithms, concerns to how imprecision is handled. Note that if only precise bids (without indicating substitute items) are inserted in the MLDA, it behaves as a set of independent auctions since no bids are suitable to win in more than one lane. So, the simplest case for a MLDA occurs when all bids are precise. In this case lanes can be cleared independently because no bid in a lane can displace a bid in another lane. Contrarily, when there are imprecise bids (substitute preferences) that can be inserted in more than one lane the efficiency obtained by the auction will be directly affected by the placement of the imprecise bids. Thus, when an imprecise bid B' is placed in a lane, the bid B' may prevent a precise bid B" from winning in that lane. In order to avoid any possible inefficiency, an imprecise bid may win only in the lane where the social welfare is maximised.

4.1 Bid Insertion

When a bid B is inserted in a MLDA, the list of possible lanes L where the bid can be placed is given. B can only affect the social welfare of any of the lanes in L. Many different situations can happen:

- B can displace a winning bid in any of the lanes in L
- B can make a current losing ask in any of the lanes in L be promoted.
- B cannot win in any of the lanes in L.

The condition to be maintained is that the Social Welfare is kept at a maximum level, so that the choice of any of those situations is given by the condition that maximises the current social welfare. So the question now is how to calculate the social welfare without having to compute it for every lane. In our algorithms we keep pointers to the current Lowest Losing Ask (LLA), the Lowest Winning Bid(LWB) and the Highest Losing Bid(HLB). Pointers are updated each time a bid or ask is inserted. Using these pointers, we can easily find the lane where the social welfare is maximised by inserting B. If a bid has to be displaced and substituted by B, the bid to be displaced will be the Lowest Winning Bid, because we want to maximise the welfare and the LWB is the worst bid that can be displaced. A displaced Bid, instead of being inserted in the losers list directly is kept in a buffer that we call the PendingLosingBids queue. Contrarily, when the situation that maximises social welfare is the one that corresponds to a promotion of a losing ask, the best choice will be the Lowest Losing Ask because there is not another ask, that if promoted, the social welfare can be improved. The pointer to the Highest Losing Bid is used to determine whether B can be discarded directly and inserted to the PendingLosingBids queue. As long as bids arrive, and there are no asks, they are directly inserted in the PendingLosingBids queue. This queue acts as a buffer and maintains bids sorted in a descending order. As we will see later the ask insertion will take advantage of this queue.

4.2 Ask Insertion

We considered that asks cannot be imprecise because it does not make sense for a seller to offer imprecise time-specified services. A seller will always indicate the specific service that it is selling including its specific time slot. The ask insertion algorithm also has to maintain the invariant, that is, keep the social welfare at a maximum level at each ask insertion. When an ask S is inserted in a lane several things need to be checked. If S is higher than the current Lowest Losing Bid, there is nothing to do and S has to be inserted in the Sout list in its lane. In any other situation S has a chance to be a winner. To make S win a currently losing bid B that can be placed in L has to be found. Once B is found ,it can be be promoted as winning bid in L and matched with S. Due to our invariant, the selected bid has to be such a bid that keeps the social welfare at a maximum level. This condition holds when the selected bid is the Highest Losing Bid that can be placed at L. The HLB that can be placed at L can be found either in the PendingLosingBids queue or in the Bout list in a lane. The algorithm selects the highest out of the possible bids. Even in that situation another condition has to be checked. It can happen that a current winning ask S' in a lane L' is higher than S and in L' there is a winning bid B' that can be moved to L. In this case, the social welfare would be improved by displacing B' to L and removing S' from the Sin in L'. To determine the best choice the following condition is checked:

whenever $B-S <= B'-S'+B-S$ it is better to promote the Highest Losing Bid that can be inserted at L. Otherwise, it is better to displace B' from L' to L and remove S' from the Sin in L'. Finally in the case that there are no suitable bids to be inserted/displaced at L, S is directly inserted in Sout in L.

4.3 PendingLosingBids Queue

The PendingLosingBids queue is a data structure that keeps bids organised in lanes. For each lane a decreasing sorted list is kept. Whenever a bid is inserted in the pending queue, a pointer to the bid is inserted in each lane where the bid can be placed. It offers functionalities to get the maximum bid out of a set of lanes. As introduced before, the PendingLosingBids queue is used to keep bids that at insertion time are not able to win or have been discarded. As asks arrive bids are removed from PendingLosingBids queue. Whenever no more asks arrive, bids are kept in the queue and considered to be losing bids.

```
Data: A Bid Bnew and $P_{lanes}$ the list of lanes where Bnew can be inserted
Result: A bid is inserted in its corresponding lane or in the pending queue
begin
    LWBordered ← List of the LWB ordered increasing
    LLAordered ← List of the LLA ordered increasing
    HLBordered ← List of the HLB ordered decreasing
    initialize (LWBmin,LLAmin,HLBmax)
    if LWBordered $is\ \emptyset \wedge$ LLAordered $is\ \emptyset \wedge$ HLBordered $is\ \emptyset$ then
        insertPendingSortedLosingBids (bnew)
        return
    if LLAordered $is\ \emptyset \wedge$ Bnew $\geq$ HLBmax then
        if Bnew $>$ LWBmin then
            displaceLWB (Bnew, LWBmin)
        else
            insertPendingSortedLosingBids (Bnew)
    else if $\neg$ LLAordered $is\ \emptyset \wedge$ Bnew $\geq$ HLBmax then
        if LLAmin $\geq$ Bnew $\geq$ LWBmin then
            displaceLWB (Bnew, LWBmin)
        else if  LLAmin $\geq$ LWBmin $\geq$ Bnew then
            insertPendingSortedLosingBids (Bnew)
        else if LWBmin $\geq$ Bnew $\geq$ LLAmin then
            promoteLLA (Bnew, LLAmin)
        else if LWBmin $\geq$ LLAmin $\geq$ Bnew then
            insertPendingSortedLosingBids (Bnew)
        else if Bnew $\geq$ LLAmin $\geq$ LWBmin then
            displaceLWB (Bnew, LWBmin)
        else if Bnew $\geq$ LWBmin $\geq$ LLAmin then
            promoteLLA (Bnew, LLAmin)
        else
            return
    else if  $\neg$LLAordered $is\ \emptyset \wedge$ Bnew $\leq$ HLBmax then
        insertPendingSortedLosingBids (Bnew)
    else if LLAordered $is\ \emptyset \wedge$ Bnew $\leq$ HLBmax then
        insertPendingSortedLosingBids (Bnew)
    else
        return
end
```

Algorithm 1: Bid Insertion algorithm

```
Data: Snew the ask to be inserted in L the target lane.
Result: Inserts and ask.
begin
    if LLA (l) is $\emptyset \vee$ Snew $<$ LLA (l) then
        checkIfCanBePromoted (Snew, L)
    else if Snew $\geq$ LLA (l) then
        insertSout (Snew,L)
    else
        return
    end
end
```

Algorithm 2: Ask insertion algorithm.

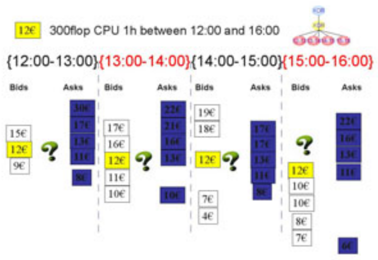
4.4 MLDA Clear and Quotes

The clearing operation is simple and straightforward. Clearing can be done, sequentially or in parallel because dependencies amongst lanes have been removed at insertion time. Thus, the clearing process matches highest bids with lowest asks by just traversing Sin and Bin. Price quotes are offered by every lane and are also easy to find, the lowest winning bid is the Bid quote and is given by the head of Bin that also corresponds to the Mth price as stated by the literature. (M+1)st price corresponds to the ask quote and is given by the head of Sin.

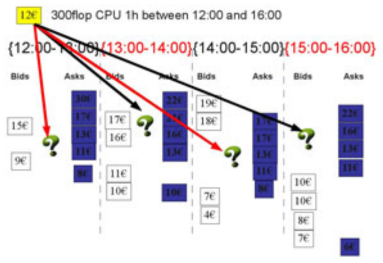
5 Implementation and Experiments

MLDA has been implemented and a set of experiments have been carried out. The aim of the simulation was twofold: first validate that MLDA provides optimal[1] efficiency and second compare its computational efficiency with multiple instances of single item double auctions. To carry out the experiments a set of data sets have been generated. Several distribution functions have been used to generate random data. The distribution functions used were derived from several experiments found in the literature [Phelps(2007), Phelps(2006), Mills and Dabrowski(2008)]. Uniform distribution of ask prices are motivated by the assumption that costs of services are also uniformly distributed. Bid prices have been generated using different distribution functions, Binomial distribution and Uniform distribution. Furthermore, bids and asks were also distributed across time slots following different distribution functions. Distributing bids and asks across different time slots using different distribution functions enabled us to experiment the effects of non-uniform supply/demand across time slots. Every experiment describes the reason for the distribution function used and the aim of the intended evaluation.

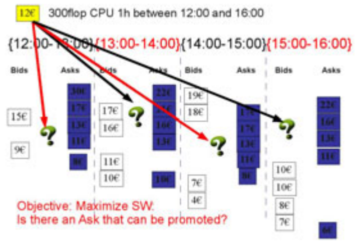
[1] There is no other allocation that improves the obtained social welfare.

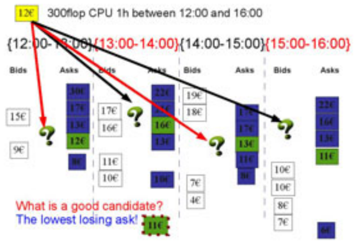


(a) Where to place the bid?

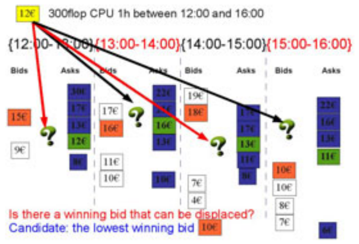


(b) The bid have to be placed in one of these lanes.


12€ 300flop CPU 1h between 12:00 and 16:00
{12:00-13:00}{13:00-14:00}{14:00-15:00}{15:00-16:00}
Objective: Maximize SW.
Is there an Ask that can be promoted?
(c) Objective Maximize SW. Is there an ask that can be promoted?



(d) The best candidate ask is the lowest losing ask because if promoted SW will be maximized.


12€ 300flop CPU 1h between 12:00 and 16:00
{12:00-13:00}{13:00-14:00}{14:00-15:00}{15:00-16:00}
Is there a winning bid that can be displaced?
Candidate: the lowest winning bid
(e) But.. if there is a currently winning bid that if displaced SW can be improved...

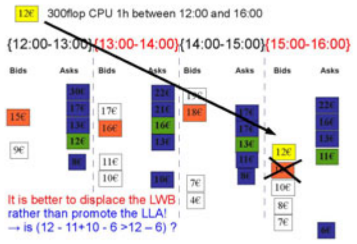

12€ 300flop CPU 1h between 12:00 and 16:00
{12:00-13:00}{13:00-14:00}{14:00-15:00}{15:00-16:00}
It is better to displace the LWB
rather than promote the LLA!
→ is (12 - 11+10 - 6 >12 – 6) ?
(f) For this example is better to displace a winning bid.

Fig. 2 Process of inserting a bid into a MLDA instance. Selection of the best candidate lane to place the bid.

```
Data: Lanes a Map of lists representing the Multi-Lane catalog of bids
Result: Returns a Map of lists. Each lists contains the pairs of matched bids
        and asks.
begin
    Map $\kappa \leftarrow \emptyset$
    for $l \in Lanes$ do
        $\alpha \leftarrow emptyset$
        $\alpha \leftarrow$ getMatchings (l)
        put ($\kappa$,$\alpha$)
    end
    return $\kappa$
end
```

Algorithm 3: Clear algorithm.

5.1 Experiment A: Economical Efficiency

The experiment aimed to evaluate the economical efficiency obtained by the MLDA. Economical efficiency has been defined as the social welfare that the mechanism provides given a certain input. Social welfare has been computed as:

$$SW = \sum(Bids) - \sum(Asks)$$

5.1.1 Experiment A Setting

In order to evaluate the economical efficiency of the MLDA we aimed to make a comparison with another well-known double auction, the k-Double Auction (k-DA). The JASA k-DA framework [Phelps(2006),Phelps(2007)] implementation has been used to make a comparison with our implementation of the MLDA. The JASA k-DA was based on the 4Heap Algorithm implementation presented by Bao et. al [Bao and Wurman(2003)]. Experiments where conducted in a dual core T9500, 2.5GHz with 4Gb of Memory.

5.1.2 Experiment A Description

The following tables describe the experiments carried out.Table 1 summarizes our first set of experiments with MLDA. The experiment consisted of the generation of a set of 1000 bids and 600 asks[2]. Several auction instances were created, one MLDA auction instance for four lanes, and three sets of four 4HeapDoubleAuction

[2] The amount of bids and asks has been determined after several experimentation with different quantities of bids and asks, starting from 10 bids and 5 asks to 3000 bids and 1500 asks. 1000 bids and 600 asks have been considered a significative amount to evaluate MLDA. Of course, the choice have also been corroborated by other related work. Phelps thesis experiments with bid and asks sets of 30 to 1000 units. Mill and Dabrowski present different experiments using between 250 and 5500 processor requirements. In their homogeneous experiment, buyers required 500 processors and sellers offered 500 processors.

Table 1 Experiment A.1 setting

Experiment A.1				
Attribute	**MLDA**	**4HDARR**	**4HDAU**	**4HDAN**
Repetitions	100	100	100	100
Lanes	4	4 instances	4 instances	4 instances
Bids	1000 (all inserted at the MLDA instance)	1000 (distributed round robin at lanes)	1000 (Uniformly distributed at lanes)	1000 (Following a Binomial PDF with n=3 and p=0,4)
Asks	600 (Uniformly distributed amongst lanes)	600 (Uniformly distributed amongst lanes)	600 (Uniformly distributed amongst lanes)	600 (Uniformly distributed amongst lanes)

instances from JASA framework [Phelps(2006),Phelps(2007)]. Each set was referred as 4HDARR, 4HDAU and 4HDAN respectively.

The experiment marked all the bids for all the lanes (imprecise bids) of the MLDA, so as to indicate that the bids were for substitutable items. Afterwards all the asks were inserted and uniformly distributed amongst the lanes. The time of computation was measured. The computation time for all the experiments measured the time taken to initialise the instance of the auction, the time taken to insert all the bids and all the asks and then finally the time to clear the auction. Furthermore, the social welfare and the number of matches where computed at the finalisation of the experiment.

For the cases of 4HDARR, 4HDAU and 4HDAN, the same experiment was carried out. For 4HDARR, bids where inserted in a round robin fashion in each lane instead of being described as substitutable for all lanes as in the case of MLDA. This means that Bid 1 was placed in 4Heap Auction instance representing lane one, Bid 2 was placed in the 4Heap Auction instance representing lane 2, Bid 3 in the auction representing the 3rd and so on... The asks were inserted following a uniform distribution amongst the lanes. The experiment measured the time taken to compute the initialisation of the four instances of the 4HeapAuction, as well as the time taken to insert the bids and asks and clear the auction. 4HDAU and 4HDAN behaved accordingly but with the difference that bids where inserted following a uniform distribution and a binomial distribution respectively.

For the four experiments the same set of bids and asks where used in order to avoid divergences due to randomisation. The experiments were repeated 100 times, re-generating the bids and asks in each experiment.

Table 2 presents the setting for the second experiment. Experiment A.2 set the distribution of asks following a Binomial PDF with n=3 (the number of lanes (0 to 3)) and p=0.24 to centre it near to lane 1. Binomial distribution was used since we wanted to evaluate a distribution of asks where not all lanes had the same probability but some of them were preferred amongst others.

Table 2 Experiment A.2 setting

Experiment A.2				
Attribute	**MLDA**	**4HDARR**	**4HDAU**	**4HDAN**
Repetitions	100	100	100	100
Lanes	4	4 instances	4 instances	4 instances
Bids	1000 (all inserted at the MLDA instance)	1000 (distributed round robin at lanes)	1000 (Uniformly distributed at lanes)	1000 (Following a Binomial PDF with n=3 and p=0,4)
Asks	600 (Following a Binomial PDF with n=3 and p=0,24)	600 (Following a Binomial PDF with n=3 and p=0,24)	600 (Following a Binomial PDF with n=3 and p=0,24)	600 (Following a Binomial PDF with n=3 and p=0,24)

5.2 Results Analysis

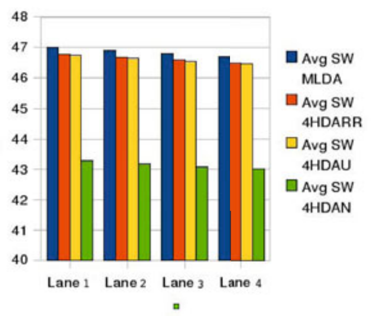
5.2.1 Experiment A.1 Results

Experiment A.1 aimed to evaluate the Social Welfare obtained by MLDA in the setting described above. By construction MLDA keeps social welfare optimal[3] so the expected results were that it achieves the best social welfare amongst other auctions. As can be seen in Figure 3.a the average social welfare amongst the 500 experiments in each lane is the highest for MLDA. 4HDARR and 4HDAU achieve close to MLDA social welfare, since the bids and asks are distributed proportionally in each lane. However, the benefit of MLDA is that the bids are always placed in the best option when 4HDARR and 4HDAU are restricted to placing them in one specific lane. 4HDAN achieves a worse social welfare due to an imbalance in the distribution between bids and asks.

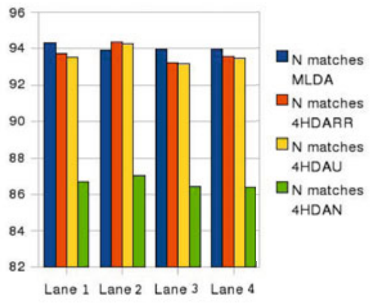
The same expected results were achieved when considering the number of matches. As can be seen in Figure 3.b MLDA achieved the highest number of matches, either per lane or in total. For the 4HDAN, we can see that the lane with highest probability achieves the highest number of matches, but it still obtains the worst results.

Figure 3.c compares MLDA with 4HDA. The Figure presents the gain in % of MLDA when compared to 4HDARR,4HDAU and 4HDAN. The compared metrics have been economical efficiency, i.e. improvement in the obtained social welfare and number of matches provided. MLDA is slightly better, 0.5%, than 4HDARR and 4HDAU when asks are distributed following a uniform distribution amongst the lanes. In these cases, the improvement obtained by MLDA is not very significant due to the distribution of bids in 4HDARR and 4HDAU, which places bids almost uniformly across the lanes. As the asks and bids were placed following the same type of distribution, the number of matches per lane in 4HDARR and 4HDAU are

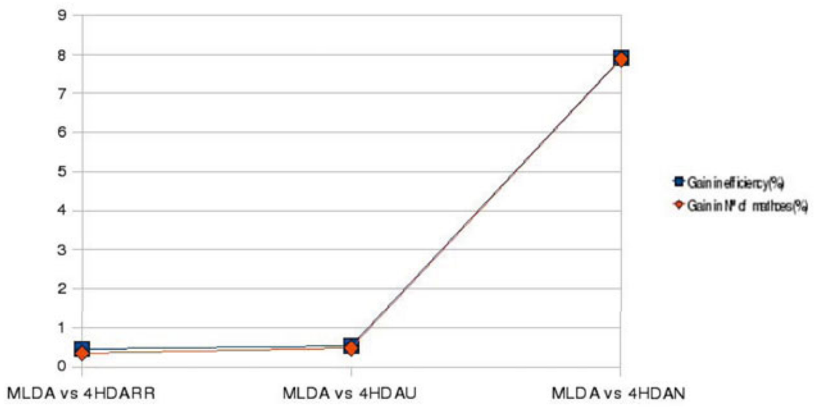
[3] See Section 4 for the description of the algorithm.



(a) Compared average social welfare per lane. Y-axis indicates Social Welfare.



(b) Number of matches per experiment and per lane. Y-axis indicates number of matches.



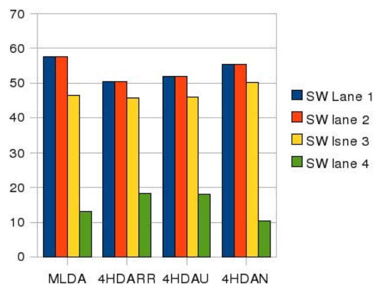
(c) MLDA compared to 4HDA. Y-axis indicates improvement in %.

Fig. 3 Results of the experiment A.1

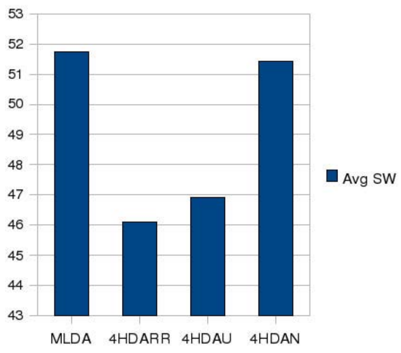
close to the allocation obtained by MLDA. However, 4HDAN behaves poorly due to the distribution of the bids being MLDA at least 8% better than 4HDAN. This of course can be attributed to the imbalance between the distribution of the asks and bids that leaves lanes 1 and 4 with a lower number of bids, which reduces the overall welfare and number of allocations. In this Figure a direct relationship can also be seen between economic efficiency and the number of allocations. It can be deduced that the gain in efficiency is directly proportional to the gain in the number of allocations.

5.3 *Experiment A.2 Results*

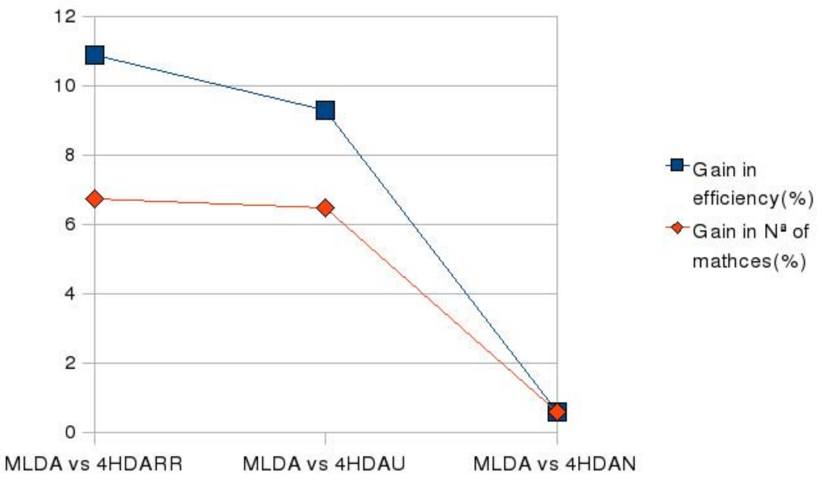
Like experiment A.1, experiment A.2 aimed to confirm that the MLDA achieves the best Social Welfare when compared to multiple auction instances. This second experiment distributed the asks following a binomial distribution as previously described. The reason for such distribution is to see the effects of heterogeneous supply and demand distribution amongst the lanes.



(a) Compared average social welfare per lane. Y-axis indicates Social Welfare.



(b) Average Social Welfare per experiment. Y-axis indicates Social Welfare.



(c) MLDA compared to 4HDA. Y-axis indicates improvement in %.

Fig. 4 Results of the experiment A.2

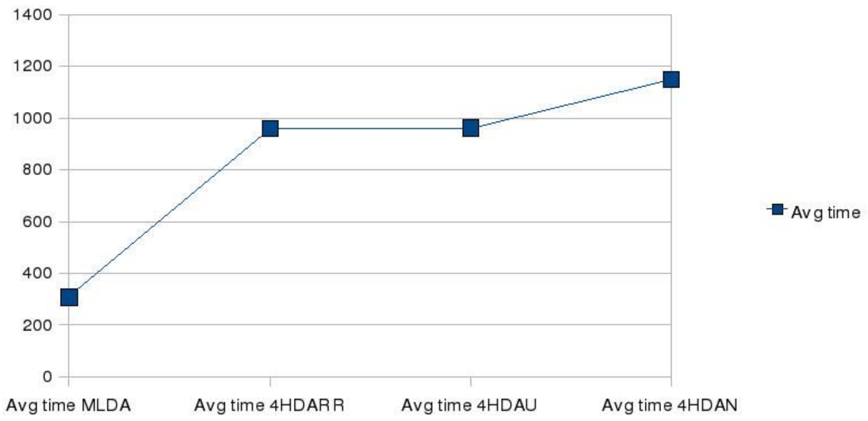
Figure 4.a shows that lanes 1 and 2 achieved higher social welfare compared to 3 and 4 and this, of course, is due to the distribution of the asks. Compared results show that MLDA gets the highest social welfare for lanes 1 and 2 while 4HDAN achieves the best social welfare in lane 3. Regarding average social welfare amongst lanes (see Figure 4.b), MLDA achieves the best social welfare followed by 4HDAN. 4HDAU and 4HDARR achieve the worst social welfare due to the binomial distribution of asks and their almost equiproportional distribution of bids.

Finally, Figure 4.c shows the gain in efficiency (in terms of social welfare) in % when comparing MLDA with other auctions. MLDA compared with 4HDARR shows that MLDA is 11% more efficient than 4HDARR, 8% more efficient than 4HDAU and even 1% more efficient than 4HDAN, when playing in its optimal situation.

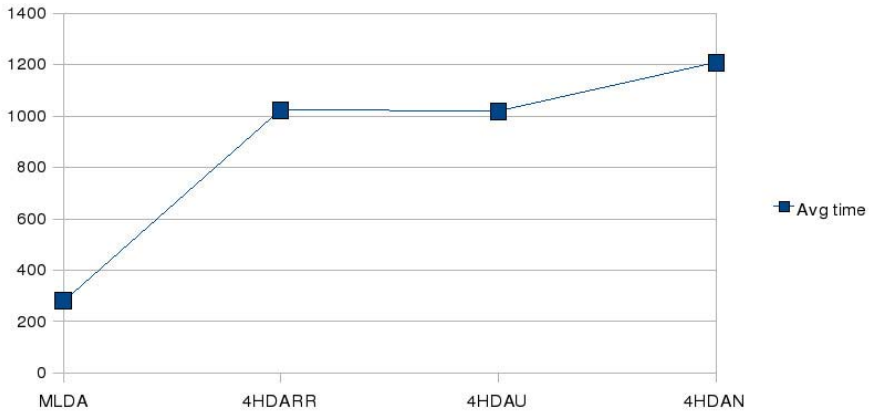
5.4 *Experiment B: Computational Efficiency*

Experiment B aimed to evaluate the computational efficiency of MLDA. Experiments A.1 and A.2 have been used to obtain experimental data concerning the computational efficiency of the different auctions. Time of computation in milliseconds has been used to determine the computation efficiency of the auction. For each experiment, the same operations were measured. MLDA was measured once the data had been generated. Measurements started at MLDA instance creation and subsequent bid insertion. After the insertion of the asks and the clearing of the measurements ended. 4HDA (in any of their variants) were also measured after data generation. Namely, instance creation, bid and ask insertion and subsequent clearing were measured. Results show that 4HDA settings take almost four times longer to finish the computation. We attribute the extra time to initialise different instances, as well as its management. Moreover, we observe that when bids in MLDA are for a restricted set of lanes the time of computation is reduced due to diminutions of the search space of the MLDA algorithm (i.e. as more bids are restricted to one lane, the lower the number of bid searches in other lanes, besides, as the number of bids for all lanes reduces, there is less search time in the overall search space). In addition, we want to point out the effects of the PendingLosingBids queue that keeps losing bids in it instead of inserting them into the Bout structures at each lane. This also shortens the time of computation for MLDA. Figure 5.a shows the time taken to carry out experiment A.1. It shows that MLDA is almost 4 times better than other 4HDA. The 4HDAN is the worst case, produced apparently by a higher number of bids in one instance in respect to others that produce a higher number of bid displacements.

Figure 5.b shows similar results when asks are inserted following a binomial distribution.



(a) Time of computation for experiment A.1. Y-axis indicates time of computation in milliseconds.



(b) Time of computation for experiment A.2. Y-axis indicates time of computation in milliseconds.

Fig. 5 Results of the experiment B

5.5 Experiment C: Scale Sensibility

The test aimed to analyse the scalability in number of lanes of the MLDA. It is supposed to obtain a linear increment of computation time as the number of lanes increases linearly. Moreover, we aimed to prove that the relationship between the performance of MLDA and multiple instances of 4HDA are maintained.

The experiment has been defined as follow:

The results obtained can be seen in Figure 6. It can be seen that our expectations were met. MLDA keeps being 2 to 3 times faster as multiple auction instances of

Table 3 Experiment C.1 setting

Experiment C.1				
Attribute	**MLDA**	**4HDARR**	**4HDAU**	**4HDAN**
Repetitions	100	100	100	100
Lanes	1, 5, 10, 15, 20, 24 lanes	1, 5, 10, 15, 20, 24 instances	1, 5, 10, 15, 20, 24 instances	1, 5, 10, 15, 20, 24 instances
Bids	1000	1000 (distributed round robin at lanes)	1000 (Uniformly distributed at lanes)	10000 (Following a Binomial PDF with n=3 and p=0,24)
Asks	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)

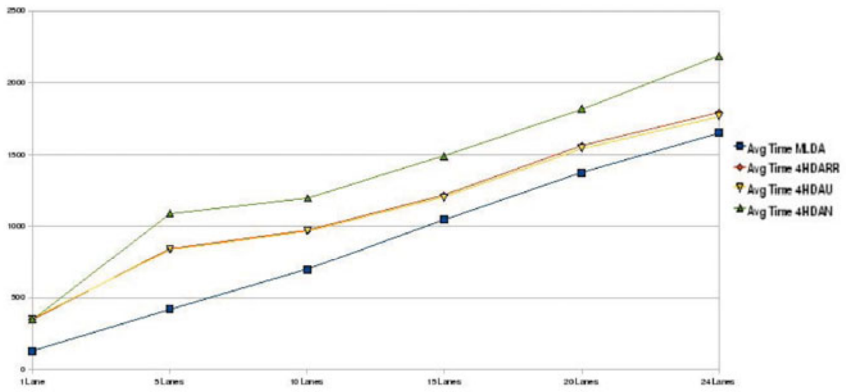


Fig. 6 Compared average execution time for different number of time slots (lanes). Y-axis indicates time of execution in milliseconds.

4HDA. We measured the scalability of MLDA in comparison to the others. One of the measures taken is the time increment between the different numbers of lanes for each experiment. On average, MLDA increases the time of computation to around 250 ms, being almost the same as in other 4HDA experiments. We conclude that there is a linear increment of time as the number of lanes increases.

5.6 *Experiment D: Price per Time Slot*

Up until now, the experiments did not consider the effects of bid prices in the final result of the auction. In this experiment we want to measure some of the effects of price distribution in the allocation provided by the auction, as well as the price

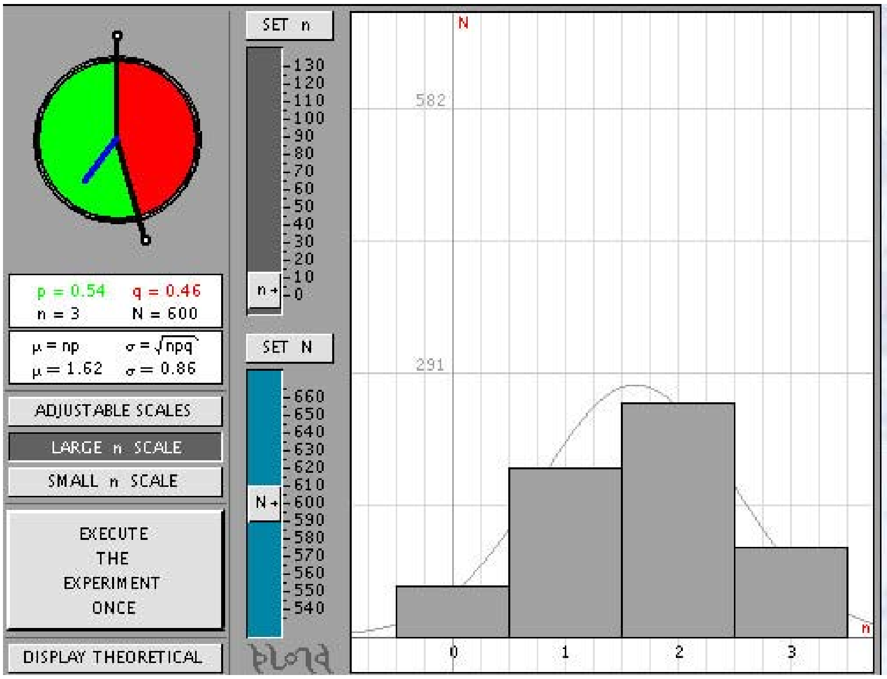


Fig. 7 Binomial Distribution for asks in experiment D.2

per time slot. Previous experiments distributed bids following different distribution functions amongst the lanes, however, prices were generated following a uniform distribution for both sellers and buyers. It is not clear right now, what is a correct distribution for bid prices in a market. It is not reasonable to assume that prices for bidders are distributed uniformly, but more probably bid prices can follow a Normal distribution or even a Pareto distribution for a specific lane. Contrarily for asks, it seems reasonable to assume that their prices follow a uniform distribution, due to the fact that costs can be assumed to be homogeneously distributed.

Experiment D.1 aims to evaluate the final price per time slot obtained by MLDA and 4HDA and verify that the results obtained by MLDA remain optimal. Furthermore, it aims to analyse the consequences of optimality[4] to the price per time slot. It is not clear whether the mechanisms that achieve best social welfare will achieve higher prices per time slot.

Moreover, prices per lane cannot be assumed to be uniformly distributed, but they should follow a long tailed distribution, such as a left shifted Normal distribution or a Pareto distribution. The reason for that is that almost all bidders want their services to be allocated as soon as possible, so as nearer the time, the higher their willingness to pay for the service. For that reason the experiment will place bids with higher prices close to the first offered time slot. Thus, the willingness of buyers

[4] In social welfare terms.

Table 4 Experiment D.1 setting

Experiment D.1				
Attribute	**MLDA**	**4HDARR**	**4HDAU**	**4HDAN**
Repetitions	100	100	100	100
Lanes	4 lanes	4 instances	4 instances	4 instances
Bids	1000 for all lanes with prices distributed in a N(0.5,0.2)	1000 distributed round robin at lanes and prices distributed in a N(0.5,0.2)	1000 Uniformly distributed at lanes and prices distributed in a N(0.5,0.2)	10000 Following a Binomial PDF with n=3 and p=0,24 and prices distributed in a N(0.5,0.2)
Asks	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)

to pay more to obtain earlier services will be simulated. In this experiment prices will be determined following the k-pricing rule that computes a non-discriminatory price for all matches in the lane. The k value has been set to 0.5 to distribute welfare in an equitable way amongst buyers and sellers. Prices have been computed as:

$$p = k \times (pM + 1) + (1k) \times pM$$
$$s.t. 0 \leq k \leq 1$$

where the pMth price and pM+1st price are the price quotes for the lane. Experiment D.2 will discuss some of the results obtained by the experiment described in the following table:

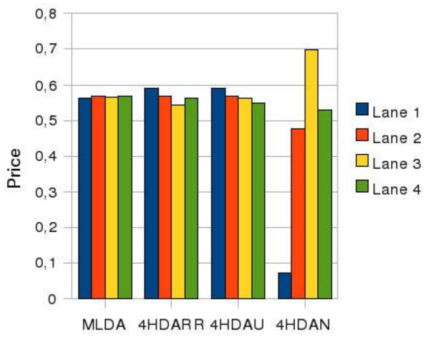
Table 5 Experiment D.2 setting

Experiment D.2	
Attribute	**MLDA**
Repetitions	100
Lanes	4 lanes
Bids	500 for lanes 1 and 2 with prices distributed in a N(0.65,0.2) and 500 for lanes 3 and 4 with prices distributed in a N(0.5,0.2)
Asks	600 (Following a Binomial PDF with n=3 and p=0.54 distribution amongst lanes)

5.7 Experiment D.1 Results

As described in Table 4, the experiment set a Normal distribution of ask prices centred at 0.5 and with a standard deviation of 0.2, this makes prices appear normally distributed between 0 and 1. It seems more reasonable to assume that prices in a market are distributed following a non-homogeneous distribution where a high percentage of bidders express similar valuations for the time slot rather than a uniform distribution of prices between 0 and 1 as in previous experiments. Figure 8 shows the distribution of the average prices per time slot (per lane) achieved by MLDA and other experiments with 4HDA. In that configuration, asks have been distributed uniformly across lanes and this is important to understand the results obtained. MLDA, achieves almost a constant price per time slot at around 0.57 due to the capacity of MLDA to place bids in the lane where the social welfare is maximised and because of the uniform distribution of asks that makes that the best configuration consists of distributing bids homogeneously across lanes. 4HDARR and 4HDAU achieve similar results as MLDA with slight more variations due to their incapacity to adapt the demand to the offer in an optimal manner. Their prices per time slot are around 0.57 and 0.55. Worse results are obtained by 4HDAN, due to the way bids are distributed. Lane 1 achieved a very low price per time slot, due to the low demand received in that time slot. Contrarily, Lane 3 obtained a high price per time slot (0.7) that can be attributed to the higher demand for that time slot. We conclude that there is a direct relationship between the demand for a time slot and the final price achieved in that Lane when the offer is fixed.

On the right-hand of Figure 8 the average prices per time-slot paid by every transaction can be seen. 4HDAN achieves higher prices due to a lower number of matches. However, 4HDAN achieves the worst Social Welfare which confirms that higher prices per time slot does not indicate better mechanism efficiency, in fact higher prices are a result of supply and demand balance for each lane. Figure 9 shows the number of matches per lane obtained by the fourth experiments. As already stated, MLDA achieves the highest number of matches while 4HDAN the worst due to the distribution of bids and asks.



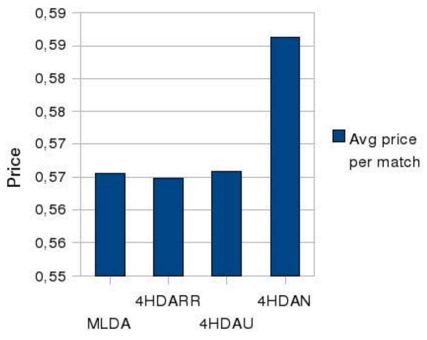


Fig. 8 Distribution of prices per lane and prices per time slot. Y-axis indicates price.

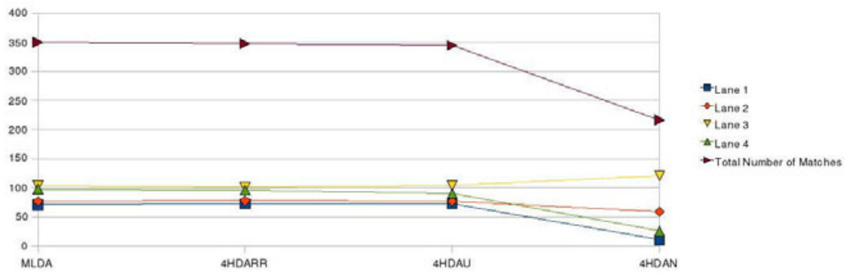


Fig. 9 Matches per lane and total number of matches. Y-axis indicates number of matches.

5.8 Experiment D.2 Results

As described in Table 5 the experiment aimed to price the first two time slots (lanes) higher to simulate the willingness of buyers to allocate services as soon as possible. Prices for lanes 1 and 2 have been generated following a Normal distribution with a mean of 0.65 and a standard deviation of 0.2. Prices for lanes 3 and 4 have been calculated using a Normal distribution with mean of 0.5 and standard deviation of 0.2. Asks where distributed non-uniformly across lanes (following a binomial distribution with n=3 and p=0.54), establishing a major number of asks for lanes 2 and 3.

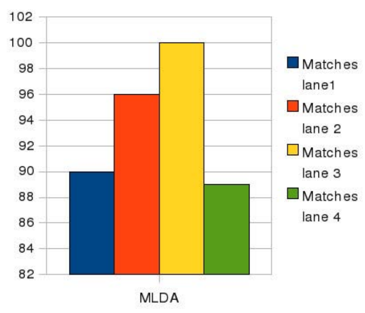
Figure 10.a shows the number of matches per lane obtained by the MLDA. Lane 3 is the lane that obtains more matches due to the distribution of the asks. Lane two obtains a lower amount of matches, even though prices are higher, due to a lower quantity of asks in that lane.

Figure 10.b shows the distribution of the social welfare generated by MLDA amongst lanes. The asks distribution guides the number of matches per lane and is the most significant factor for the final allocation provided by MLDA. Finally, the effects of higher prices in lanes 1 and 2 can be seen in Figure 10.c. Even though lane 2 has a higher number of matches, which should mean a lower price per time slot, it achieves a similar price per time slot to lane one that achieves the maximum due to a short offer and higher bid prices.

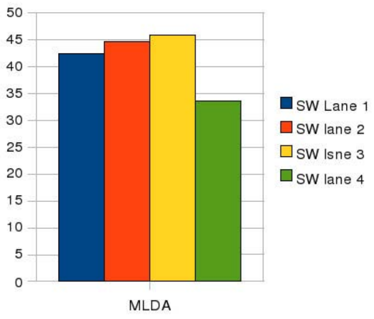
It can be concluded that asks distribution constrains the type of allocation provided by MLDA since asks can only win in the lane for where they have been submitted. It can also be pointed out that the higher the bid price, the higher the number of matches when asks have uniform prices.

5.9 Experiment E: Memory Usage

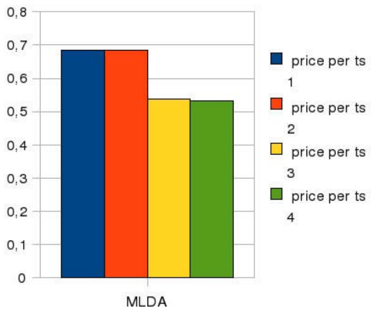
Experiment E aimed to evaluate the memory consumption of MLDA when compared to any of the 4HDA implementations used so far. The experiment aims to analyse the overall amount of memory used during the process of bids and asks insertions. It is clear that the amount of memory used will depend on the size of



(a) Matches per lane. Y-axis indicates number of matches.



(b) Social Welfare per lane. Y-axis indicates social welfare.



(c) Price per time slot. Y-axis indicates price.

Fig. 10 Results of the experiment D.2

the data structures used in the implementation. For this experiment is not important the total amount of memory used by both instances, but the relation between the amount of memory used. What is important to know is whether MLDA uses less, more or the same amount of memory than 4HDA, as well as the ratio of the difference. Table 6 summarises the experiment setting that consisted of 500 iterations of an experiment that inserted 1000 bids and 600 asks to an instance of every one of the evaluated auctions.

Memory usage has been measured during the insertion of the bids and asks. Measures were taken just before instantiating the auction and just after finishing the insertion of the last bid and ask. The system's garbage collector has been called before the first measurement and just after the last measurement. The amount of memory used has been calculated as a difference between the initial amount of memory and the final amount of memory.

Table 6 Experiment E.1 setting

Experiment E.1		
Attribute	**MLDA**	**4HDA**
Repetitions	500	500
Lanes	4 lanes	4 instances
Bids	1000 for all lanes	1000 round robin amongst lanes
Asks	600 (Following a Uniform distribution amongst lanes)	600 (Following a Uniform distribution amongst lanes)

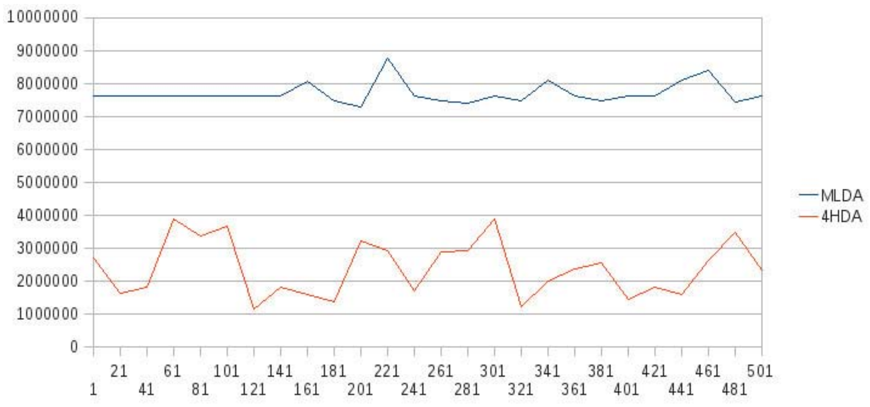
5.10 Experiment E Results

The results of the 500 experiments can be seen in Figure 11.a. MLDA spends nearly 3 times more memory than 4HDA in almost all experiments. Slight variations of memory usage at each experiment can be attributed to the runtime. These variations are more significant at 4HDA since four instances of auction objects are maintained. However, the general line is well defined and the relation is almost constant. Figure 11.b shows the ratio between MLDA and 4HDA. MLDA uses in average 3.23 times more memory than 4HDA, which is attributed to the PendingLosingBids queue that maintains a pointer for each bid to the lanes where a bid can be placed, and to the sorted lists used to maintain the different quotes across lanes.

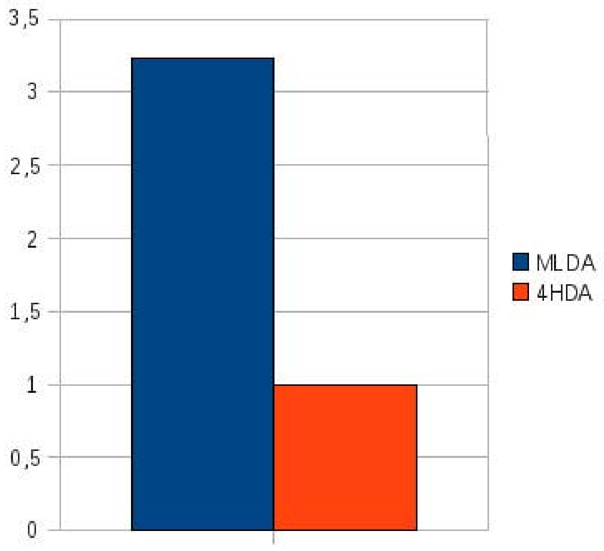
MLDA achieves better computational performance, as demonstrated in previous experiments, at the expense of using more memory. MLDA improves in computational efficiency two to three times to any 4HDA, while it used 3 more times of memory. However, MLDA provides the benefit of dealing with substitute preferences that any other 4HDA can deal with.

6 Related Work on Resource Allocation Frameworks

The term market mechanism is encountered in connection with problems of distributed resource allocation. In the context of markets it refers to a structure of economic organisation that helps to shape outcomes. Intuitively [Nisan and Ronen(2001)] a mechanism solves a problem by assuring that the required allocation occurs when agents choose their strategies to maximise their own utility. A mechanism also needs to ensure that the agent reported utilities are compatible with the algorithm implementing the mechanism. Economic mechanisms propose a procedure by which a set of resources may be distributed amongst the different participants and a scheme for pricing of the traded resources. The allocation is constrained by the preferences of the participants expressed in monetary terms. This section gathers the most significant auction based resource allocation frameworks found in the literature.



(a) Memory usage during 500 experiments. Y-axis indicates the amount of memory used in bytes.



(b) Compared memory usage between MLDA and 4HDA. Y-axis indicates the relation of the amount of memory required.

Fig. 11 Results of the experiment E

6.1 Auction Markets for Single Type of Resource

SPAWN [Waldspurger et al(1992)Waldspurger, Hogg, Huberman, Kephart, and Stornetta] was designed to tap into unused and wasted cycles in networked servers. Each participating server runs an auction process to trade the CPU time in fixed time-slices. Spawn uses a sealed bid second-price auction, known as the Vickrey auction. Vickrey auction is incentive compatibility, i.e. the best strategy that the bidders may practise is to reveal their true valuations. This system is not generalised to multiple resources and multiple resource units - considering time-slices as a resource unit; this implies an auction for each time-slice.

Placek et. al. [Placek and Buyya(2006)] present a trading platform for storage services. The platform implements a centralised storage exchange that implements a double-auction; the exchange accepts sealed offers from providers and consumers and periodically allocates trades by employing an algorithm that maximises surplus, that is, the difference between the consumer's price and the seller's cost. Double auctions are adapted to the trading of a single type of homogeneous resource. These have the benefit of reducing communication costs (single bids) and with suitable pricing policies are also incentive compatible.

6.2 Auction Markets for Multiple Types of Resources

The combinatorial auction model has received a lot of attention in recent years; to address trading multiple resource types in bundles; this has two implications: (i) prices are expressed for bundles and (ii) a bundle, if allocated, should be completely satisfied.

Chun et. al. [Chun et al(2006)Chun, Ng, Albrecht, Parkes, and Vahdat] present a resource discovery and allocation system where users may express preferences using a bidding language supporting XOR bids, and at most one of the preferences is allocated. Multiple resources may be requested to a central auction server that clears periodically. Resource requests are for fixed durations of time and users may specify the time ranges. A greedy algorithm clears the combinatorial auction. This algorithm privileges execution time over efficiency of allocation - bids are ordered by decreasing values where the value is obtained by dividing the bid price by the product of total number of resources and the duration of request.

Schwind et. al. [Schwind et al(2006)Schwind, Gujo, and Stockheim] present an iterative combinatorial auction that maximises seller revenues. Bids are presented as a two-dimensional matrix; one dimension represents the time in fixed time slots and the other dimension the resources (CPU, Disk, and network). The auction server executes periodically and invites bids from the participants. Shadow prices are calculated for individual items (resource) and the buyers are requested to iterate on their bids based on the current estimation of prices. The clearing algorithm is implemented as a linear program optimising the revenue. Prices are calculated using the approach presented by [Kwasnica et al(2005)Kwasnica, Ledyard, Porter, and DeMartini] âĂŞ the prices are the dual solution to the primal Linear Programming Problem (LP).

Schnizler and Neumann [Schnizler(2007)] present a multi-attribute combinatorial auction that maximises the surplus - the difference between the buyer's price and seller's cost. Resources are traded in fixed time-slots and buyers send XOR bids specifying the quality and the number of time-slots within a specified time range.

The Vickrey pricing policy is applied to provide incentive compatibility. Simulation results show that the allocation problem is computationally demanding but feasible in the case where the number of participants and bids are reasonably small.

Bellagio [AuYoung et al(2004)AuYoung, Chun, Snoeren, and Vahdat] is a market-based resource allocation system for federated distributed computing infrastructures. Users specify interest on resources in the form of combinatorial auction bids. Thereafter, a centralised auctioneer allocates resources and decides payments for users. The Bellagio architecture consists of resource discovery and resource market. For resource discovery of heterogeneous resources, Bellagio uses SWORD [Albrecht et al(2004)Albrecht, Patterson, and Vahdat]. For resource market, Bellagio uses a centralised auction system, in which users express resource preferences using a bidding language, and a periodic auction allocates resources to users. A bid for resource includes the sets of resources desired, processing duration and the amount of virtual currency which a user is willing to spend. The centralised auctioneer clears the bid every hour. The resource exchange in the current system is done through a virtual currency. The virtual currency is the amount of credit a site has, which is directly determined by the site's overall resource contribution to the federated system. Bellagio employs Share [Chun et al(2004)Chun, Ng, Albrecht, Parkes, and Vahdat] for resource allocation in order to support a combinatorial auction for heterogeneous resources. Share uses the threshold rule to determine payments. Once the payment amount of each winning bid has been determined by the threshold rule, the winning bidders receive resource capabilities after being charged the appropriate amount.

Even though the above approaches indicate the computational complexity of combinatorial auctions, they nevertheless are important demonstrators. Combinatorial auctions are important mechanisms for Grid resource markets since typically Grid applications need to allocate resources in bundles.

6.3 Proportional Sharing Markets for Divisible Resources

Market based proportional sharing models are one of the most popular approaches in problem-solving environments. Basically this approach consists of allocating users a percentage of the resource that is proportional to the amount of the bid submitted by the user. This may be considered as a fair model of allocation and is typically employed in cooperative environments employed in systems where resources are considered as divisible.

Tycoon [Kevin Lai and Fine(2004)] is a system designed for time-sharing networked nodes, such as in PlanetLab; an environment where users of resources are also providers of resources. Resource nodes execute an auction process to which users may send their bids. Tycoon implements a proportional sharing [Kelly(1997)] auction where resources are shared and each user is attributed a capacity proportional to its bid. Users are price-anticipating in that the ratio that they receive is a proportion of their bid over the sum of all bids for a given resource and users may anticipate the effect of their bids on the clearing price. Each user has a utility function; a weighted sum of the resource fraction that it receives from each node. Users bid to those nodes that maximise their utility. This system is intrinsically decentralised as the maximisation of the utility is done locally by each user. This system is appropriate for divisible usage of a CPU, an assumption that may not be acceptable to a wide range of applications.

6.4 Decentralised Markets for Single Type of Resources

Peer-to-peer auctioning has emerged as a new computing paradigm to decentralise the auction processes. Many systems address the decentralised auctioning issue for different reasons like those of scalability, fault-tolerance, redundancy, load distribution and autonomy, amongst others. With respect to scalability issues, most of the existing systems make use of a DHT structured overlay [Ratnasamy et al(2001)Ratnasamy, Francis, Handley, Karp, and Schenker, Stoica et al(2001)Stoica, Morris, Karger, Kaashoek, and Balakrishnan, Castro et al(2002)Castro, Druschel, Kermarrec, and Rowstron, Ghodsi(2006)].

PeerMart [Hausheer and Stiller(2005)] distributes brokering in an auction based allocation mechanism. Auctioneers, rather than being a single broker, are formed by a set of peers which synchronise to clear a double auction market. Consumers and providers are distributed in the Pastry overlay network [Castro et al(2002)Castro, Druschel, Kermarrec, and Rowstron]. Broker sets are formed by some nodes in the overlay and the double auction they implement clears continuously. For each allocation, brokers synchronise their information in order to determine who the winners are and to avoid malicious peers. Synchronisation is carried out by broadcasting the lowest selling requests, the highest buying bid and any matching bid to the rest of the brokers in the broker set. Decisions are taken by a simple majority. Broker set peers maintain a distributed shared state through broadcasting information and decisions are taken when all peers in the broker set have all the information.

Tamai et.al. [Tamai et al(2005)Tamai, Shibata, Yasumoto, and Ito] use CAN [Ratnasamy et al(2001)Ratnasamy, Francis, Handley, Karp, and Schenker] to build a market architecture. They propose distributing peers into different sub-regions and assigning a responsible broker for each of them. Sell requests and buying bids are sent to any known broker and they are forwarded until they reach the broker responsible for the offer. If the broker has a buying bid (sell requests) that matches the received sell request (buying bid) the allocation is made. However, if a buying bid and a sell request are received by different brokers respectively, they cannot be matched. To solve that problem, Tamai et.al. propose replicating buying bids and sell requests in multiple brokers. The replication introduces more communication amongst peers. When a replica matches a sell request with a buying bid it has to verify that the original bid (request) has not been matched by sending a message to the original replica. Once a buying bid and a sell request are matched, a replica deletion message has to be sent to all replicas.

As proof of this concept, in the paper by Despotovic et.al. [Despotovic et al(2004)Despotovic, Usunier, and Aberer], a simple approach is presented. The paper presents a mechanism for pricing and clearing continuous double auctions in a peer-to-peer system. The main feature is that consumers and providers broadcast bids for resources. Every buyer has the incentive to trade with the announcer of the lowest sell request that the buyer observed. Similarly, any seller would want to trade with the announcer of the highest observed bid. Prices in each trading operation are set to the average price between the bid and the sell request. Since peers do not have a global view of all the trading operations that occur in the system (when a trading operation is made between a buyer and a seller, we cannot assume that they will communicate their price to the rest of the bidders), prices are updated when a peer observes a bid or request from another peer. Clearly, the solution is not scalable and there are no guarantees that the information reaches all the peers. However, the mechanism can be useful for market implementations that do not require optimality and efficiency.

Esteva et.al. [Esteva and Padget(1999)] make use of a ring topology to distribute one side (English, Vickrey, and Dutch) auction processes amongst a set of brokers (called interagents). Interagents are mediators amongst buyers and the auctioneer and are responsible for receiving bids and clearing the auction. The clearing algorithm is based on the leader election algorithm [Lynch(1996)]. In short, the algorithm is based on finding the bidder with the highest bid, which will be the leader. The aim of introducing interagents is to reduce centralisation and the work of the auctioneer. However the authors do not indicate where interagents are executed and if they can be sellers or buyers participating in the auction. The authors point out that interagents have to be robust and introduce security measures against malicious peers. However, they do not introduce any security measure.

Atzmony and Peleg [Atzmony and Peleg(2000)] propose a set of algorithms for clearing English auctions in a distributed manner. They assume an underlying communication network represented by a complete n-vertex graph. Vertices represent the nodes in the network and every two vertices are connected with an edge that represents a bidirectional connection. Auctions are hosted by a subset of nodes in the graph. In the paper, they formally present a set of algorithms to clear an English auction in a distributed manner. The algorithm requires a static set of participants that join the auction before it starts. They also propose some enhancements to allow dynamic participants. Each auction is executed in several rounds until only one bidder remains. At the end of each round, new bidders are allowed to join the auction. Their asymptotic approach only finishes if no more bidders join the auction and all bidders except one resign. Although they formally verify the algorithm, the paper does not present any results on the performance and communication costs of the algorithm in a real network.

Several conclusions can be derived from those systems. Distribution of auction processes is costly. Decentralised auctions usually map one item to one responsible broker. In some other approaches, auction decentralisation can be compared to a way of sorting distribution problems, which can then be solved. The nature of auctions require complete information (i.e. the total set of bids is required) in order to determine the winner set and consequently decentralisation can only come when multiple items are traded assigning different responsible brokers for each item. In our work decentralisation is attained by short-lived market instances since we consider co-allocation and substitute allocation important aspects that need to be adressed.

7 Conclusions

In this paper a novel variant of the well-known Double Auction has been presented. The paper motivated the utility of the auction, as well as setting the context to be applied. MLDA constitutes an important contribution to the building of scalable and Cloud oriented service marketplaces. The paper presented the data structures used to design MLDA, as well as the main operations and algorithms that constitute the core of the auction. In order to evaluate the mechanism, an extensive set of experiments through simulation have been carried out. MLDA, due to its design, achieves optimal allocations in terms of Social Welfare.

The evaluation compared MLDA with another well-known implementation of the Double Auction. We tested MLDA and compared results with different configurations of the other mechanism. The results showed a better behavioural level for MLDA in all situations, due to the properties of the auction (i.e. invariant maximum social welfare). Moreover, MLDA has been compared with several double auction instances running at the same time. This experiment showed us the computational performance (i.e. time taken to execute) of MLDA when compared to a set of auctions potentially providing the same allocation. MLDA showed two to three times more computational efficiency that is MLDA is two to three times faster than multiple instances of single item double auctions.

Finally, MLDA showed to be a good candidate auction to trade time-differentiated services especially under the presence of substitute preferences. Besides MLDA can be considered a light-weight alternative to Combinatorial Auctions (CAs), being able to provide efficient allocations without dealing with the computational costs of CAs. The next step of the evaluation will compare MLDA with an instance of a Combinatorial Auction.

Acknowledgements. This work is being partially supported by the Grid4All European project under contract FP6-IST-034567 and Spanish MCYT project TIN2008-01288/TSI. This work has also been supported by the HAROSA Knowledge Community (http://dpcs.uoc.edu). Besides we feel deeply indebted with Leandro Navarro and Dirk Neumann for their helpful advice.

References

[Albrecht et al(2004)Albrecht, Patterson, and Vahdat] Albrecht, J., Patterson, D., Vahdat, A.: Distributed resource discovery on planetlab with sword. In: WORLDS - First Workshop on Real, Large Distributed Systems (2004)

[Atzmony and Peleg(2000)] Atzmony, Y., Peleg, D.: Distributed algorithms for english auctions. In: Herlihy, M.P. (ed.) DISC 2000. LNCS, vol. 1914, pp. 74–88. Springer, Heidelberg (2000)

[AuYoung et al(2004)AuYoung, Chun, Snoeren, and Vahdat] AuYoung, A., Chun, B., Snoeren, A., Vahdat, A.: Resource allocation in federated distributed computing infrastructures (2004), citeseer.ist.psu.edu/auyoung04resource.html

[Bao and Wurman(2003)] Bao, S., Wurman, P.R.: A comparison of two algorithms for multi-unit k-double auctions. In: ICEC 2003: Proceedings of the 5th International Conference on Electronic Commerce, pp. 47–52. ACM, New York (2003), http://doi.acm.org/10.1145/948005.948012

[Buyya and Venugopal(2004)] Buyya, R., Venugopal, S.: The gridbus toolkit for service oriented grid and utility computing: An overview and status report (2004), citeseer.ist.psu.edu/buyya04gridbus.html

[Castro et al(2002)Castro, Druschel, Kermarrec, and Rowstron] Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks. In: EW 2010: Proceedings of the 10th Workshop on ACM SIGOPS European Workshop: beyond the PC, pp. 140–145. ACM Press, New York (2002), http://doi.acm.org/10.1145/1133373.1133399

[Chun et al(2004)Chun, Ng, Albrecht, Parkes, and Vahdat] Chun, B.N., Ng, C., Albrecht, J., Parkes, D.C., Vahdat, A.: Computational resource exchanges for distributed resource allocation. Tech. rep. (2004)

[Chun et al(2006)Chun, Ng, Albrecht, Parkes, and Vahdat] Chun, B.N., Ng, C., Albrecht, J., Parkes, D.C., Vahdat, A.: Computational resource exchanges for distributed resource allocation (2006), citeseer.ist.psu.edu/706369.html

[Consortium(2008)] Consortium, G.: Grid4all european project (2008), http://grid4all.eu/

[Despotovic et al(2004)Despotovic, Usunier, and Aberer] Despotovic, Z., Usunier, J.C., Aberer, K.: Towards peer-to-peer double auctioning. In: HICSS 2004: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004) - Track 9, p. 90289.1. IEEE Computer Society, Washington (2004)

[Esteva and Padget(1999)] Esteva, M., Padget, J.A.: Auctions without auctioneers: Distributed auction protocols. In: Agent Mediated Electronic Commerce (IJCAI Workshop), pp. 220–238 (1999), citeseer.ist.psu.edu/497541.html

[Eymann et al(2003)Eymann, Reinicke, Ardaiz, Artigas, Freitag, and Navarro] Eymann, T., Reinicke, M., Ardaiz, O., Artigas, P., Freitag, F., Navarro, L.: Decentralized resource allocation in application layer networks. ccgrid 00:645 (2003), http://doi.ieeecomputersociety.org/10.1109/CCGRID.2003.1199427

[Ghodsi(2006)] Ghodsi, A.: Distributed k-ary System: Algorithms for distributed hash tables. PhD dissertation, KTH—Royal Institute of Technology, Stockholm, Sweden (2006)

[Haussheer and Stiller(2005)] Haussheer, D., Stiller, B.: Decentralized auction-based pricing with peermart. In: Integrated Network Management, pp. 381–394. IEEE, Los Alamitos (2005)

[Kelly(1997)] Kelly, F.: Charging and rate control for elastic traffic (1997), citeseer.ist.psu.edu/kelly97charging.html

[Kevin Lai and Fine(2004)] Kevin Lai, B.A.H., Fine, L.: Tycoon: A Distributed Market-based Resource Allocation System. Tech. Rep. arXiv:cs.DC/0404013, HP Labs, Palo Alto, CA, USA (2004)

[Kwasnica et al(2005)Kwasnica, Ledyard, Porter, and DeMartini] Kwasnica, A.M., Ledyard, J.O., Porter, D., DeMartini, C.: A new and improved design for multiobject iterative auctions. Manage. Sci. 51(3), 419–434 (2005), http://dx.doi.org/10.1287/mnsc.1040.0334

[Lai et al(2005)Lai, Rasmusson, Adar, Zhang, and Huberman] Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.A.: Tycoon: An implementation of a distributed, market-based resource allocation system. Multiagent Grid Syst. 1(3), 169–182 (2005)

[Liu and He(2007)] Liu, Y., He, H.C.: Multi-unit combinatorial auction based grid resource co-allocation approach. In: International Conference on Semantics, Knowledge and Grid, vol. 0, pp. 290–293 (2007), http://doi.ieeecomputersociety.org/10.1109/SKG.2007.26

[Lynch(1996)] Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publishers Inc., San Francisco (1996)

[Mills and Dabrowski(2008)] Mills, K.L., Dabrowski, C.: Can economics-based resource allocation prove effective in a computation marketplace? Journal of Grid Computing 6, 291–311 (2008)

[Neumann et al(2007)Neumann, Stößer, Anandasivam, and Borissov] Neumann, D., Stößer, J., Anandasivam, A., Borissov, N.: Sorma - building an open grid market for grid resource allocation. In: Altmann, J., Veit, D. (eds.) GECON 2007. LNCS, vol. 4685, pp. 194–200. Springer, Heidelberg (2007)

[Nisan and Ronen(2001)] Nisan, N., Ronen, A.: Algorithmic mechanism design. Games and Economic Behavior 35, 166–196, 613 (2001)

[Phelps(2006)] Phelps, S.: Web site for JASA (Java Auction Simulator API) (2006), http://www.csc.liv.ac.uk/sphelps/jasa/

[Phelps(2007)] Phelps, S.: Evolutionary mechanism design. Ph. D thesis, University of Liverpool, U.K. (2007)

[Placek and Buyya(2006)] Placek, M., Buyya, R.: Storage exchange: A global trading platform for storage services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 425–436. Springer, Heidelberg (2006)

[Radhanikanth and Narahari(2009)] Radhanikanth, G.V.R., Narahari, Y.: Reverse combinatorial auction&#45;based protocols for resource selection in grids. Int. J. Grid Util. Comput. 1(2), 109–120 (2009), http://dx.doi.org/10.1504/IJGUC.2009.022027

[Ratnasamy et al(2001)Ratnasamy, Francis, Handley, Karp, and Schenker] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM 2001: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 161–172. ACM, New York (2001), http://doi.acm.org/10.1145/383059.383072

[Schnizler(2007)] Schnizler, B.: Mace: A multi-attribute combinatorial exchange. In: Jennings, N., Kersten, G., Ockenfels, A., Weinhardt, C. (eds.) Negotiation and Market Engineering, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 06461 (2007), http://drops.dagstuhl.de/opus/volltexte/2007/1009 [date of citation: 2007-01-01]

[Schnizler and Neumann(2007)] Schnizler, B., Neumann, D.: Combinatorial exchanges for coordinating grid services. SIGecom Exch. 7(1), 65–68 (2007), http://doi.acm.org/10.1145/1345037.1345054

[Schwind et al(2006)Schwind, Gujo, and Stockheim] Schwind, M., Gujo, O., Stockheim, T.: Dynamic resource prices in a combinatorial grid system. In: CEC-EEE 2006: Proceedings of the The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, p. 49. IEEE Computer Society, Washington (2006), `http://dx.doi.org/10.1109/CEC-EEE.2006.37`

[Stoica et al(2001)Stoica, Morris, Karger, Kaashoek, and Balakrishnan] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160 (2001), `citeseer.ist.psu.edu/stoica01chord.html`

[Tamai et al(2005)Tamai, Shibata, Yasumoto, and Ito] Tamai, M., Shibata, N., Yasumoto, K., Ito, M.: Distributed market broker architecture for resource aggregation in grid computing environments. In: CCGRID 2005: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005), vol. 1, pp. 534–541. IEEE Computer Society, Washington (2005)

[Waldspurger et al(1992)Waldspurger, Hogg, Huberman, Kephart, and Stornetta] Waldspurger, C.A., Hogg, T., Huberman, B.A., Kephart, J.O., Stornetta, S.: Spawn: A distributed computational economy. IEEE Transactions on Software Engineering 18(2), 103–117 (1992)