Towards decentralized resource allocation for collaborative peer to peer learning environments

Xavier Vilajosana, Daniel Lázaro, Joan Manuel Marquès, Angel A. Juan
Universitat Oberta de Catalunya
{xvilajosana,dlazaroi,jmarquesp,ajuanp}@uoc.edu

Abstract

Collaborative e-learning virtual communities use virtual learning environments provided by the university or tools that are available in Internet. Although this model has proven to work, it has limitations for adhoc and peer-to-peer groups due to the dynamic nature of its resources. A way to deal with it is by defining virtual organizations (VO) that gather the resources and interests of their members in a way that they can lend resources to or borrow them from other VO. This paper presents DyMRA, a decentralized resource allocation system based on markets that allows inter-VO resource allocation. DyMRA is specially designed for collaborative peer-to-peer environments, where the autonomy of participants and its decentralized nature requires the capacity to dynamically reallocate resources and services that manage the overall system. DyMRA is built on top of LaCOLLA, a peer-to-peer middleware that allows a group of users to share resources in a collaborative manner. We present the design, architecture and validation of our proposal.

1 Introduction

Nowadays nobody doubts that E-Learning is an effective and useful way to learn, as demonstrates the success of many virtual universities – the Open University of Catalonia (http://www.uoc.edu), with its almost 40.000 students, is an example of it – or the fact that, in traditional universities, many subjects are taught totally or partially in a virtual manner. Collaborative e-learning has also deserved a lot of attention and many solutions have been proposed. In either case, the resulting virtual communities use virtual learning environments provided by the university which helps to preserve the community notion – or using tools that are available in Internet – either for free or paying. Although this model has proven to work, it has important limitations, like the following three: learning institutions has to estimate the amount of resources needed to deal with peak situations, with the resulting over-dimension of the system; different systems (e.g. belonging to different faculties) do not share resources, which may result in having in the same university some systems overloaded at some periods of the year while other systems, at this period, are underutilized; and that ad hoc collaborative groups are only partially supported, because the tools and resources available to them are restricted to university capacity and polices.

A way to deal with those limitations is by defining virtual organizations (VO) that gather the resources and interests of their members in a way that they can lend resources to or borrow them from other VO. Examples of VO can be a faculty department or a group doing a collaborative activity. Whilst the resources within a VO may be sometimes insufficient to satisfy QoS requirements under unexpected load surges, high level of dynamicity of its members or unpredictable failures, computers that belong to another VO may have surplus bandwidth, storage and computations resources. This opens challenging opportunities to promote inter-VO resource allocation. In other words, a VO could aggregate their surplus resources and offer them to other VOs.

In this paper we address Inter-VO allocation of resources by means of decentralized markets that promote the creation of many local ad hoc trading sites that can be accessed by any VO. Markets have proven their ability to allocate resources efficiently [12] and, more importantly, they provide mechanisms through which the need may be correctly elicited and quantified; and indeed, they promote incentives to resource owners to provide or trade their resources.

We developed DyMRA [10], a decentralized resource allocation system based on markets that allows inter-VO resource allocation. DyMRA is specially designed for dynamic and peer-to-peer environments, where the autonomy of participants to disconnect resources at any time and its decentralized nature requires the capacity to dynamically reallocate resources and services that manage the overall system.

DyMRA markets are created at will and run as services within the VO. The choice of a decentralized markets approach in the form of many local ad hoc markets is motivated by the need to deal with dynamic communities and scalability issues that would be limited by a centralized approach.

One issue not addressed in this paper due to space limitations is that of resource specification and bidding language. Many interesting approaches [6] present efficient bidding

[0]Work supported by MCYT-TSI2005-08225-C07-05, Grid4All(IST-2006-034567) and TIN2007-68050-C03-01.

languages that fulfill our requirements. For the purpose of this paper, we consider resources as processing time, storage, and applications that provide a stateless service, like an efficient codec or a parallelizing compiler. We deal with heterogeneity by using standard interfaces, implemented as WS, to access the resources. These belong to a VO and we assume that they can be disconnected or fail at any time. This dynamic behavior introduces a complexity that, added to the decentralized behavior of markets, forced us to design a system that allows us to decouple services from physical resources. Furthermore, we had to pay special attention to availability guarantees.

Another important aspect in a system that allows allocation of external resources is that of defining a medium of exchange, namely a virtual currency. A virtual currency facilitates the transfer of services or resources. Furthermore, the virtual currency rather than storing value acts as a token that simplifies preference elicitation and provides incentives to share resources while facilitating trading without incurring in real payment. In some cases, it may be interesting that virtual currency could be translated into real money. In some environments, though, this might not be desirable. For example, in the case of VOs formed inside a university, resources should be shared without needing to exchange real money. Hence, we use virtual currency as a means to quantify the resources a VO can allocate, leaving the option to translate this into real money for the environments where it's appropriate.

DyMRA is built on top of LaCOLLA[1] [11, 9]. LaCOLLA is a peer-to-peer middleware that allows a group of users scattered across the Internet to share resources in a cooperative manner and that allows the deployment of stateless services using the resources provided by the members of the VO. LaCOLLA guarantees that services deployed are always available (if enough resources are provided). Therefore, DyMRA components are deployed as services in LaCOLLA middleware.

2 Related work

Economic based resource allocation within the context of Virtual Organizations, Grid Computing and large scale peer-to-peer systems has been extensively studied [5]. However, as far as we know, the issue of addressing inter-VO resource allocation is an emergent field of study. Recently in [1] a novel architecture for inter-VO resource allocation in Grids is presented. Their proposal is suited between a centralized and a decentralized approach and proposes a configurable mediator process (executing within the VO) that allocates resources from external providers. Our approach goes one step beyond and we provide, on one hand, transparent reallocation of market services in a dynamic environment, and, on the other hand, we rely on the members of collaborating communities to mutually provide resources.

Another feature that we addressed is that of dynamic deployment of services. Other systems, like Snap [8], also perform this task in a decentralized and self-organized way. Snap nodes form a Distributed Hash Table (DHT) which stores the code and data of the services. Replicas of a service are created on demand and stopped when demand decreases.

[1] Available at http://dpcs.uoc.edu/lacolla/.

3 Architecture

The architecture of DyMRA consists of a series of components which are in charge of the trading process. These components are:

- **Prospector**: when external resources are needed, it is in charge of finding a suitable market and obtaining the desired resources.
- **Seller**: it is in charge of offering the aggregated surplus resources of the VO in a suitable market.
- **Pool service**: it controls the access of the VO members to the external resources acquired by the VO, acting as a mediator.
- **Sale Handler**: it controls the external access to a set of resources sold to another VO, acting as a mediator.
- **Accounting service**: it monitors the resources available in a VO. Following a policy determined by the VO, it starts the acquisition of external resources or the cession of own resources to other VOs when convenient.
- **Market**: it mediates the trading of resources between VOs.
- **Market Directory**: Contains an index of existing markets and their locations.

The system is built upon a middleware called LaCOLLA [11] which allows a small group of computers connected through internet to participate in collaborative activities and sharing their resources (i.e. provides virtualization of resources), while tolerating high levels of dynamism. This middleware also allows the deployment of services within a VO (or group) [9]. When a service is deployed, the system guarantees that it will always be available, placing it in a suitable location chosen among the resources of the VO, and reinstantiating it in case of failure.

The components of DyMRA are deployed as services inside a VO (except the Market Directory), and can, hence, be reallocated when its current location fails or disconnects, keeping the functionality available. The communication between VOs is done through markets, which are also services, existing in a specific VO. To access a market, it must be discovered through the Market Directory (MD). Markets contact the MD to publish their location and characteristics, and the MD keeps them as a soft state. In case a market ceases to exist, the MD will delete the information about it after its time-to-live (TTL) expires. If a market is reallocated, it will inform the MD of its new location.

The MD is not part of a VO, but an external service which is known and can be accessed by all groups. Its implementation is out of the scope of this paper, but there are many possibilities. It could be a centralized index, but it could also be implemented in a decentralized way if each VO deployed a ”MD node” service, and each one of these services act as a node of a DHT, thus distributing the information stored among the VOs. Anyway, this doesn't affect the design of our architecture.

4 Trading process

To help understand the functionality of each of the components presented and the overall behavior of the system, we will explain in detail how the trade of resources is done at the buyer VO and at the seller VO (shown at fig. 1).

Buying resources

1a. The Accounting service detects that the resources of a certain type (e.g. storage) available in the VO are below a certain threshold defined by the VO policy. According to a given policy, it determines the resources needed and other factors such as the price that should be paid for them. With this information, it contacts the Prospector and asks it to acquire such resources.

2a. The Prospector looks for a suitable market in the Market Directory.

3a. The Market Directory sends the Prospector a list of markets which suit the specified needs.

4a. The Prospector chooses one of the markets of the list. In case that there is no suitable market, it proceeds to the creation of a new one. Once it has the adequate market located, the Prospector sends its bid. A generic bid describes the type of resource to bid for, the price per unit offered and the number of units required amongst others.

Selling resources

1b. The Accounting service detects that the resources of a certain type (e.g. storage) available in the VO are above a certain threshold defined by the VO policy. According to a given policy, it determines that these resources can be leased to another group, and fixes the price that should be paid for them. With this information, it contacts the Seller and asks it to sell the surplus resources.

2b. The Seller looks for a suitable market in the Market Directory.

3b. The Market Directory sends the Seller a list of markets which suit the specified needs.

4b. The Seller chooses one of the markets of the list. In case that there is no suitable market, it proceeds to the creation of a new one. Once it has the adequate market located, the Seller sends its offer.

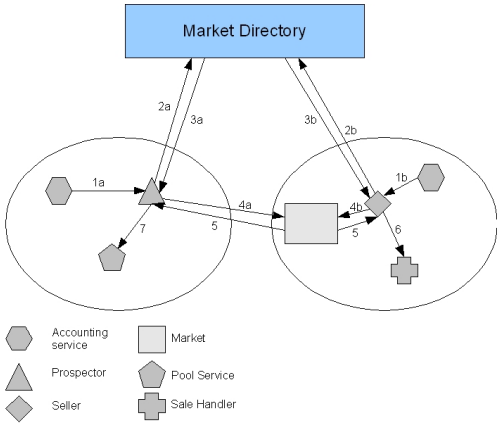

Figure 1: Interaction among components in the trading process

Agreement

5. The market makes an agreement between the buyer and the seller. A scheduled double auction is used to match winning bids and offers. The winners are selected by calculating the price where supply balances demand and matching the highest buy bids above the price with the lowest sell offers below the price. After this, it notifies the sale to both the Prospector and the Seller, by sending them the Agreement. This is explained in depth in section 5.

6. The Seller starts a Sale Handler, which is deployed in its VO. This Sale Handler keeps the information about the leasing conditions, and mediates the use of the resources according to these conditions.

7. The Prospector informs the Pool service of its group about the resources bought and the agreement conditions, as well as the location of the Seller of the resources.

When a Prospector or a Seller finds that there is no market available that suits its needs, it proceeds to the creation of a new one. As stated before, the market is implemented as a service. Hence, the component (Prospector or Seller) creates a new service in its VO, which is a market with the desired characteristics. This market registers itself in the Market Directory, and therefore can be accessed by buyers or sellers from outside the VO.

Once the agreement has been made, the buying VO can start using the external resources. Whenever a client needs to use a resource, the system checks the VO policies to determine whether it must depend only on local resources or should use external resources. In the latter case, the client contacts the Accounting service, which checks the resources currently available to the VO. Following the VO's policy, it determines what resources the client must use, whether these are internal or external. Access to external resources is made through the Pool service, which contacts the Seller to obtain the current location of the Sale Handler that manages the specific agreement.

5 Agreement and Payment

One issue addressed in DyMRA is that of agreement and contract specification and its relation with payment. In DyMRA groups make use of virtual currency that is managed by the Accounting service. At initialization each group has a certain quantity of virtual currency that may be spent on acquiring other external resources. Besides, groups may increase their amount of currency by selling their unused resources to other groups. The following subsection present how DyMRA deals with agreements and manages payment of its transactions.

5.1 Agreement

The agreement specification process is carried out by the Market that creates an agreement object. The Agreement specifies the acquired resources, the lease time, the price and the contact point (IP address and port), that is the Seller location for the case of the Agreement sent to the Prospector and the Prospector location for the case of the Agreement sent to the Seller . Besides, the Agreement also includes the final price determined by the market's pricing policy. Once the agreement has been created, it is send to both Prospector and Seller in order to either start payment protocol or to start the leasing of resources. That choice may depend on the payment policy configured for that trade.

The agreement has been specified as an XML schema extension for Agreement, which is applied, but not limited, to standard languages for job submission like WS-Agreement[3], JDL-GLUE[2] or the JSDL [4]. JSDL and JDL provide semantics for web services description but do not address the description of Agreement. Contrarily, the WS-Agreement addresses the Agreement specification but it is placed in a more generic level than our approach which can be included as a part of it.

The following code snipped presents our XML schema extension for the agreement and contract specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xsd:complexType name ="agreement Type">
 <xsd:sequence>
   <xsd:element ref ="resource"minOccurs ="1"maxOccurs ="1"/>
   <xsd:element ref ="time" minOccurs ="1"maxOccurs ="1"/>
   <xsd:element ref = "price"minOccurs ="0"maxOccurs ="1"/>
   <xsd:element ref ="partnerLocation" minOccurs ="0"maxOccurs ="1"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name ="resourceType">
 <xsd:sequence>
   <xsd:element name="ResourceDescRef" type="xsd:IDREF"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name ="leaseType">
 <xsd:sequence>
   <xsd:element name="startTime" type="timeType"/>
   <xsd:element name="endTime" type="timeTypet"/>
   <xsd:element name="slotSize" type="xsd:int"/>
   <xsd:element name="nbSlotTime" type="xsd:int"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name ="locationType">
 <xsd:sequence>
   <xsd:element name="address" type="xsd:string"/>
   <xsd:element name="port" type="xsd:int"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name ="timeType">
 <xsd:sequence>
   <xsd:element name="day" type="xsd:int"/>
   <xsd:element name="hour" type="xsd:int"/>
   <xsd:element name="minute" type="xsd:int"/>
   <xsd:element name="second" type="xsd:int"/>
   <xsd:element name="millis" type="xsd:int"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:element name ="Agreement"type ="agreementType"/>
<xsd:element name ="resource"type ="resourceType"/>
<xsd:element name ="time" type ="leaseType"/>
<xsd:element name ="price" type ="xsd:double"/>
<xsd:element name ="partnerLocation" type ="locationType"/>

<redefine schemaLocation="./agreement.xsd">
<!-- redefinition of Agreement -->

<xsd:complexType name="pricingPolicyType">
 <xsd:choice>
   <xsd:element name="payBefore" type="xsd:boolean"/>
   <xsd:element name="payAfter" type="xsd:boolean"/>
 </xsd:choice>
</xsd:complexType>

<complexType name="Contract">
 <complexContent>
  <extension base="xs:Agreement">
   <sequence>
    <element name="currency" type="xsd:string"/>
    <element name="accountId" type="xsd:string"/>
    <element name="credentials" type="xsd:string"/>
    <element name="paymentLocation" type="locationType"/>
    <element name="accessLocation" type="locationType"/>
    <element name="paymentpolicy" type="paymentPolicyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
</redefine>
```

5.2 Payment

DyMRA implements two payment policies, namely "paybefore" and "payafter". "paybefore" requires the execution of the payment protocol before the Pool Service is able to use the acquired resources. "payafter" does not require the payment until the resources have already been used.

The payment protocol is executed between the Seller and the Prospector. Once the seller has received the agreement object he creates the contract object that specifies mainly the accountID of the seller, the payment service location, i.e. the accounting service within Seller's VO, the currency used for the payment (currently in DyMRA we only use one type of currency but we devised the schema to support different types of currency).

Three messages constitute the payment protocol:

1. *requestPayment*: The Seller creates the contract indicating its accountID, the location of the accounting service of the VO, the payment policy and the credentials to access the resources only for the case of a "payafter" payment policy.

2. *payment*: When the Prospector receives the contract, he determines whether it has to send the payment to the accounting service specified in the contract, or contrarily he can get access to the resource before payment. In this case, the contract includes the credentials to access to the acquired resource.

3. *paymentAcknowledgement*: When the Seller is notified by the accounting service within its VO that the payment has been done he sends the contract again with the credentials to access the resource as well as the location of either the resource or the SaleHandler responsible of that resource (only for the case of "paybefore" payment policy).

As state before, the payment protocol can be done before accessing to the resources, in this case, once the Prospector has the credentials to access the resource he communicates

them to the Pool Service within its VO who will be responsible of accessing the resources. For the case of "payafter" payment policy, the Prospector communicates the Pool Service the credentials and the access point to resources. Once the lease expires, the Pool Service notifies to the Prospector that sends the *payment* to the corresponding accounting service.

6 Validation

This section presents an implementation of the proposed mechanism and its first validation. These preliminary results demonstrate the viability of our proposal and encourage us to refine it. Currently we are working on a further and exhaustive validation.

We implemented a prototype of the proposed architecture to test its usefulness. The Prospector, Seller, Pool, SaleHandler and the Market have been implemented as deployable services over the LaCOLLA middleware. The Market provides generic operations that allow different mechanisms to be implemented. For our testing purposes we developed a double auction protocol that enables buyers and sellers to submit bids for multiple units of a single resource (i.e storage capacity, cpu capacity and applications).

The Market Directory has been implemented as a centralized index, but, as mentioned above, it can be easily substituted with a decentralized approach such as a DHT. For our testing purposes, the market directory stores pairs of $< key, value >$ where the key identifies the type of traded resource and the value refers to the identifier of the market where it is traded in.

The objective of our test is to validate the trading process described above. One of the main objectives of our proposal is to provide good availability in environments of high dynamism and churn. Hence, availability has been the main focus of our tests.

First, we consider that services can become unavailable because of failure or disconnection of the node where they are executing. In these cases, though, the service will be started again in a different node. We estimated the average service migration time as a function of the messages exchanged amongst VO management components and the estimated transmission time in a peer-to-peer network [7]. We found that when a service is migrated to another component within the VO it takes 6 to 10 messages with a trip time of $100ms$ with highest probability [7], so we estimate that the service migration takes a time compresed between 0.6 and 1 second. This is the time that a service will remain unavailable in case of failure.

In order to obtain data about the availability perceived by users, we executed a process which periodically tried to buy resources, and another that tried to sell resources. The necessary services (Prospector, Pool, Seller) where active inside the VO, while there was a Market Directory available in a static location. Markets, though, according to our proposal, are created on demand. When a Prospector or a Seller wants to access a Market, but there isn't any available, it proceeds to create and activate one. When this happens, it is counted in our tests as a failed attempt. We have considered two configurations regarding the lifespan of the Markets. In the first one, Markets have been assigned a limited lifespan, after which they resolve the auction and send the results to the clients, and are stopped and deleted from the Market Directory. This implies that, periodically, a Prospector or a Seller will have to create a Market, thus decreasing the perceived availability. In the other configuration, Markets are permanently active. The expected result is that this increases the availability of the system. There is, though, a trade off between the obtained availability and the resources spent to keep the market active, so it is interesting to quantify the improvement in availability that can be achieved this way.

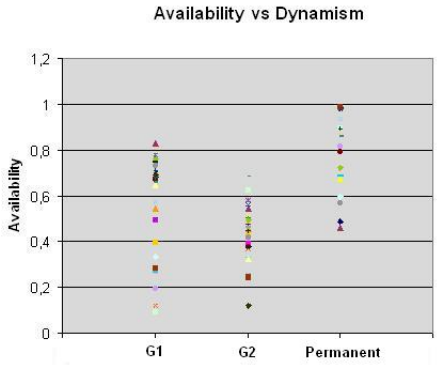

Figure 2: Availability vs level of Dynamism

The LaCOLLA middleware offers the ability to simulate users' activity and system dynamism (connections, disconnections, failures) in order to conduct tests and validate its functioning. We measured the availability of markets in function of the levels of dynamism of the system. Specifically, we evaluated two different levels of dynamism. In the less dynamic (from now on, called G1) each component had a probability of failure per iteration of 0,0005, and a probability of ordered disconnection of 0,0025. In the more dynamic of the two (G2), the probability of failure per iteration was 0,005, while the probability of disconnection was 0,008. Tests lasted 500 iterations.

The data we analyze is the number of bids that arrive to the market, in relation to the number of bids issued by the group. This depends exclusively of the mechanisms of our system, in contrast to the number of matches, which depends on supply and demand. Note once again that this number decreases in the case where markets have a limited lifespan and are created on demand, as this results in a failed access when a market must be created. That doesn't mean that, in a real situation, the bid cannot be issued. In the cases where the service is available, the bid is sent to the market immediately. In the cases where the market is not found, the bid can be send too, but after a delay that might be the time it takes to create a service, if no market existed, or the time it takes for a service to be restarted (previously estimated), if there was a suitable marktet that had failed.

Fig. 2 shows the availability (percentage of succesfully issued bids) obtained in 30 executions, for both G1 and G2, when markets have a limited lifespan. We see that, as ex-

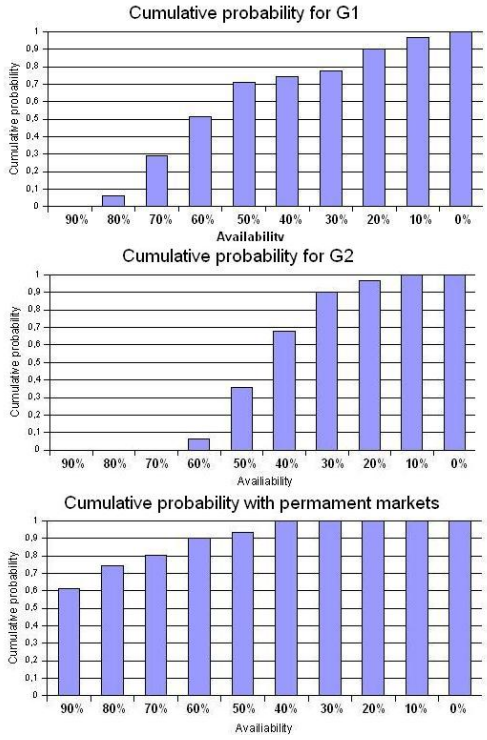

Figure 3: Cumulative probability of availability levels for G1 and G2 with limited lifespan of markets, and for G1 with permanent markets.

pected, the availability is higher in G1, decreasing in G2 because of the higher level of dynamism. It also shows the availability obtained when markets are permanently active, with the level of dynamism of G1.

Fig. 3 shows the cumulative distribution function for both G1 and G2. For G1, 50% of the executions obtain an availability of 60% or higher, which must be considered noting that markets are activated on demand, and we count it as unavailable when activation is needed. For G2, availability is low because of the high level of dynamism. When markets are permanent, instead, we obtain a much higher availability, with more than 60% of the executions obtaining an availability over 90%.

7 Conclusions

The paper proposes DyMRA, a framework for inter-VO resource allocation. The key aspect of DyMRA is that of market decentralization, that allows allocations of resources amongst different VO in spite of markets' failures. Markets and mediator components such as buyer agents and seller agents are exposed as mobile services within the VO that allows the utilization of inherently centralized mechanisms such as auctions into a decentralized environment without introducing bottlenecks or single points of failure. Furthermore the paper presents the preliminary results of evaluating our proposed architecture. Our future work includes the complete development of the DyMRA components, such as a decentralized Market Directory, and the set of mechanisms to control the access to external allocated resources. Besides, we aim to consider duration of the allocations of resources (lease times) that would permit the application of our framework in a real environment.

References

[1] N. Amara-Hachmi, X. Vilajosana, R. Krishnaswamy, L. Navarro, and J. M. Marques. Towards an open grid marketplace framework for resources trade. In *Grid computing, high-performAnce and Distributed Applications (GADA'07)(to appear), Nov. 2007.*, 2007.

[2] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Konya, M. Mambelli, J. M. Schopf, M. Viljoen, and A. Wilson. GLUE Schema Specification - Version 1.2, Dec 2005.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (ws-agreement), version 2006-09-07. Technical report, Global Grid Forum, 2006.

[4] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job submission description language (jsdl) specification, version 1.0. Technical report, Global Grid Forum, 2005.

[5] Catnets Consortium. Deliverable d3.1: Implementation of additional services for the economic enhanced platforms in grid/p2p platform: Preparation of the concepts and mechanisms for implementation (gmm), 2005.

[6] R. Cavallo, D. C. Parkes, A. I. Juda, A. Kirsch, A. Kulesza, S. Lahaie, B. Lubin, L. Michael, and J. Shneidman. Tbbl: A tree-based bidding language for iterative combinatorial exchanges. In *Multidisciplinary Workshop on Advances in Preference Handling (IJCAI)*, 2005.

[7] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann. Efficient simulation of large-scale p2p networks: packet-level vs. flow-level simulations. In *UPGRADE '07: Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, pages 9–16, New York, NY, USA, 2007. ACM.

[8] C. P. Gavalda, P. G. Lopez, and R. M. Andreu. Deploying wide-area applications is a snap. *IEEE Internet Computing*, 11(2):72–79, 2007.

[9] D. Lázaro, J. M. Marquès, and J. Jorba. Decentralized service deployment for collaborative environments. In *Proceedings of the 1st International Conference on Complex, Intelligent and Software-Intensive Systems, CISIS'07*, pages 229–234, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[10] D. Lázaro, X. Vilajosana, and J. M. Marquès. Dymra:dynamic market deployment for decentralized resource allocation. In *Proceedings of the OTM Federated Conferences and Workshops*. Springer-Verlag, 2007.

[11] J. M. Marquès, X. Vilajosana, T. Daradoumis, and L. Navarro. Lacolla: Middleware for self-sufficient online collaboration. *IEEE Internet Computing*, 11(2):56–64, 2007.

[12] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. Chun. Why markets could (but don't currently) solve resource allocation problems in systems. In *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*, pages 7–7, Berkeley, CA, USA, 2005. USENIX Association.