Distributed Resource Allocation for Contributory Systems.

Xavier Vilajosana Guillen

Estudis d' Informàtica Multimèdia i Telecomunicacions
Universitat Oberta de Catalunya

Thesis submitted for the Degree of Doctor of Philosophy in the
Universitat Oberta de Catalunya

· 2009 ·

Abstract

The thesis presents an approach to on demand capacity expansion in communities of Internet users that aggregate their resources to achieve a common objective. Such communities are emerging as forms of organization taking advantage of an increasing broadband access and computational capacity. Volunteer computing such as SETI@home, Collaborative Grids such as OurGrid and LaCOLLA, Ad-hoc and Peer-to-Peer Grids such as P-Grid and the XGrid project from Apple, Open Grids such as the addressed by SORMA and Grid4All and many other approaches of Grid Computing based on Virtual Organizations are the focus of our work. These systems are characterized by the purpose of their participants, i.e. to achieve a common objective taking advantage of the aggregation of other resources'. The cited systems, in contrast to high performance computing Grids, are open to new participants which make their behaviour unpredictable and dynamic, usually resources are connected and disconnected spontaneously. While the critical aspect of high performance Grids are computational performance, stability and availability are the main issues for the systems addressed in these work.

The thesis homogenizes the concepts of those paradigms under the term Contributory System which is used along the thesis to refer to the systems where users provide their resources to be used collectively to achieve a common objective. Resource expansion in Contributory Systems is required so as to increase the limited capacities of ad-hoc collaborative groups under unexpected load surges, temporary resource requirements or other policies defined by the objectives of the Virtual Organization that they constitute.

Four aspects are addressed through the dissertation, first it identifies the main properties and applications of Contributory Systems and motivates the need for infrastructures to enable on-demand resource expansion. This goes in the direction of Utility Computing trends which are main business lines for IT companies. Thus the thesis proposes the on demand provision of idle resources from the extremes of the Internet, other Virtual Organizations or Resource Providers to those organizations that have resource needs. In this work, resource allocation is handled by market models which provide efficient while simple mechanisms to mediate the allocation of resources. This proposal enables new emerging opportunities to Internet users to make their business on the Internet by selling their idle resources. Besides, this brings the opportunity to small communities to grow and to bring super-computing capacities to Internet end-users.

Second the thesis describes semantically Computational Resources so as to build a common knowledge about the resources in the Internet. The semantic description enables a common understanding of the nature of resources permitting the pooling and aggregation of distinct types of technologies while maintaining the same semantics. This makes applications and resource management frameworks independent of the real nature of the resources which we claim as a fundamental aspect to keep resource management independent of the dynamics and evolution of technology in computational environments such as in Contributory Systems. A semantic description permits the development of generic specifications to provide bid and offer description in computational markets.

Third, an architecture for on-demand resource expansion in Contributory Systems is presented, the architecture has been designed to provide the main functionalities to on-demand provision of resources through markets to scenarios characterized by dynamism, evolution and heterogeneity. The architecture provides the main market oriented functionalities and enables dynamic and on-demand execution of market mechanisms.

Finally, an specific Grid-oriented market mechanism is presented. The approach is motivated due to the unsuitability of current auctions to allocate efficiently time-differentiated resources (usually provided by many different resource providers) such as most of the resources in a Contributory System.

The thesis builds a roadmap to achieve flexible and decentralized resource expansion in communities where resources are shared by their participants by analysing the main scenarios where it can be applied, providing the semantics and specification to enable the description of user's requirements, proposing a flexible and configurable architecture to deal with on-demand resource expansion in Virtual Organisations and proposing an specific mechanism adapted to trade computational resources.

In the confrontation between the stream and the rock, the stream always wins. Not through strength, but by perseverance - ...

H. Jackson Brown

To my grand mother, to my parents and to my brother Ignasi
for their unconditional support and love.

Contents

List of Figures

List of Tables

Acknowledgments

I am heartily indebted to my thesis advisor, Dr. Joan Manuel Marquès for guiding me in the hard work of my thesis, for his innovative ideas, timely suggestions, feedbacks and unflinching support throughout my Ph.D candidature. I feel highly privileged to have worked with him. I owe a great debt of gratitude for his patience, inspiration and friendship. He opened the door to research and showed me the methodology of investigation and allowed me to participate in some of the work he carried for his thesis. Together we worked in LaCOLLA which allowed me to learn about peer-to-peer systems and from where most of the ideas explored in this thesis born.

I also feel deeply indebted with Ruby Krishnaswamy. For me she had been my second advisor. I would like to thank her patience, advice and experience, she trusted on me and gave me the opportunity to collaborate with her. I have learnt from her experience, wisdom and way of work which I really admire. She helped in the most difficult parts of my thesis and motivated me to investigate deeply and systematically to understand the nature of my problems. Also thanks for her valuable feedback, comments and answers.

I would also like to thank the people with whom I have received comments and advice over the course of this thesis, especially to Leandro Navarro. I sincerely appreciate the opportunity given by Leandro to participate in the Grid4All project as well as his recommendations during my work. Thanks to Angel A. Juan, for the timely clarification of my doubts and questions about statistics and maths.

My most grateful thanks to Internet Interdisciplinary Institute (IN3) that funded me during the first years of my research and gave me the opportunity to arrive here. Also to the Grid4All European project (IST-2006-034567), the Spanish government under projects MCYT-TSI2005-08225-C07-05, TIN2007-68050-C03-01 and TIN2008-01288-TSI for partially funding this research.

My thanks also to those who I collaborated with, first to Daniel Lázaro a college and a friend who I shared some of the work over LaCOLLA and DyMRA. Also to Xavi León, Rene Brunner, Daniel Stern, Josep Jorba and Pablo Chacin for all the things I learnt from them. Also my hugest greetings to Santi Caballé to sit in front of me during the long trip of the thesis. I just followed his tireless will of working.

I wish to thank to all my friends in IN3 and UOC where my research started and I hope that they can follow my path. I won't forget all those that walked together with me, specially those that are always in my heart and know that this greeting is for their unconditional friendship and love.

This thesis would never have been possible if it were not for the support of my family, especially my father, Ignasi and my mother, Carme, they have brought me here. To my grand mother also, she opened my interest to understand the nature of things, and also to my brother, Ignasi for all the time over the bike. To all of them I dedicate this long race.

Chapter 1

Introduction

This chapter introduces the context of the research themes explored in this thesis. It starts with the fundamental motivations behind decentralised and coordinated organisation of distributed systems; including resource allocation and resource management systems. The chapter thereafter provides discussion on the thesis outline and contributions. It ends with a summary of the published materials that were partially or fully utilised for compiling the thesis.

1.1 Overview

The last few years have seen the emergence of a new generation of business that operates over the Internet. This new trend is Utility computing, i.e. - the aggregation of computational resources such as computation and storage, as a metered service similar to a traditional public utility (such as electricity, water, natural gas, or telephone network). Many service providers and IT companies are moving their economies around the renting of computational services, not physically, but as an utility.(see Figure 1.1)[1] In 1961, John

[1]In April of 2007, In-Stat conducted a Web-based questionnaire that gathered data from 1003 respondents who are knowledgeable about their organizations' IT infrastructure. The survey gauges demand for utility computing services for IP telephony, CPU, storage, and bandwidth capacity. Survey results show that more than 50% of respondents from firms with over 1,000 employees either currently contract for on-demand services or, if they do not currently do so, would be very interested in doing so within the next year. For those who currently contract for on-demand computing services, saving money was cited most often as the most important reason

McCarthy from MIT stated that these emerging opportunities will arrive:

"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

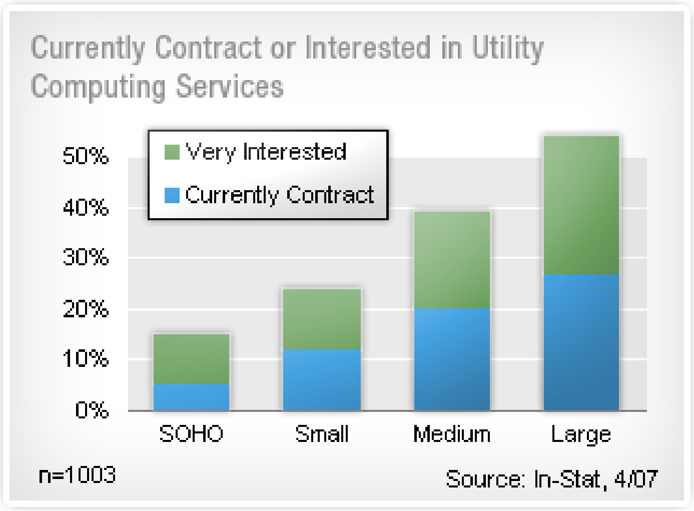


Figure 1.1: Adoption of, and interest in, on-demand or utility computing services is strongest among the largest US firms. Image extracted from http://www.instat.com.

Nowadays IT companies such as SUN, IBM, Amazon, HP and Google are offering computational resources and services for computation to third party enterprises. The access they offer is still far from end users in the Internet, since accessing computational services requires certain levels of expertise (i.e. main APIs are offered as standard web services that are not easily accessed by Internet end users). Besides these services are mainly used for computation purposes by research institutions, universities, and big companies and there are not Internet end-user applications suitable or adapted to take advantage of super-computing capacities. It is clear that the success of these business depends on its introduction and the fact that the service becomes widely used.

Many trends aimed to gather distributed and dynamic resources and services and build an infrastructure that hides the heterogeneity, distribution and scales over Internet. One of this approaches is Grid Computing [36] (see Figure 1.2). In the Grid, resources are grouped in federations under a common administrative domain, these federations are commonly referred as Virtual Organizations (VO). A Virtual Organisation integrates services and resources across distributed, heterogeneous, dynamic organisations to allow service and resource sharing when cooperating on the realisation of a joint goal. Each of these organisations is a management domain under the control of another management domain which represents the Virtual Organisation. Thus a virtual organisation is primarily a management domain that controls and coordinates the services and resources provided by others management domains to achieve a common goal.

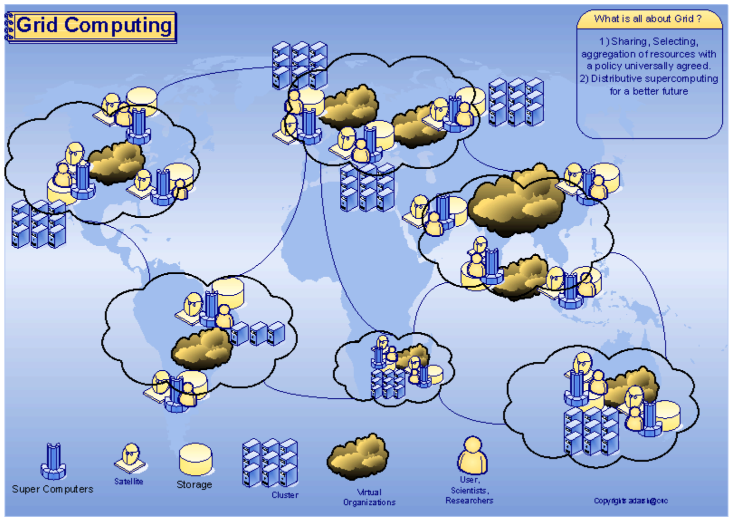

Grid Computing
What is all about Grid ?
1) Sharing, Selecting, aggregation of resources with a policy universally agreed.
2) Distributive supercomputing for a better future
Super Computers
Satellite
Storage
Cluster
Virtual Organizations
User, Scientists, Researchers
Copyright: adasi@cic

Figure 1.2: Grid Computing. Image from Grid Computation: the Fastest Supercomputer in the World (Released November 2006) by Chao-Hsu Yao

The focus of this thesis is communities where their participants are willing to collaborate, mainly Internet end-users, contributing their own resources in favour of the VO. In these systems, termed in this dissertation Contributory Systems, users provide their own resources to be used collectively by any members of the VO. The thesis deals with on-demand resource expansion and offering of resources in VOs that have limited capacities due to the limited amount of resources provided by their participants. Access to VO resources is regulated by internal policies of the VO, enabling users to access to resources when they are available. In this sense, during certain periods of time VOs may become overloaded due to dynamism of their users or due to the over-consumption of services by their participants [24]. The thesis aims to decouple the real capacities of the VO from the perceived capacities by their participants. In that sense, VOs capacity can be expanded by aggregating resources from third party providers (i.e other Internet users, Internet resource providers or other Virtual Organizations) that aim to obtain profit by offering their idle resources

It is clear that new scenarios will be possible when the grouping of resources enables the creation of VO by the user's of Internet. NGOs, schools, universities, neighbour communities, fan clubs, online gaming communities, etc. can take advantage of grouping and contributing their resources to create ad-hoc communities. However, due to the nature of the participants in the collaboration the amount of resources available would not be unlimited and consequently, opportunities to expand the capacities of their groups can contribute to the welfare of the Virtual Organization. The thesis deals with the main mechanisms and tools needed to expand VO capacity through the use of markets. As already stated the use of markets is merely for regulation purposes, that is, markets and their laws provides efficient and simple tools to regulate and control the access to scarce resources, giving the resources to those who value them most. This thesis will not address neither payment nor currency issues, that is, we are unaware of whether real currencies or virtual currencies are used, for us, they are tokens used to account and express user's willingness and utility. For simplicity, we do not also consider aspects related to open or close economies, we just suppose that VOs have their amount of tokens that are used to obtain resources from external providers.

The addressed scenario can be generalized as a set of VOs/projects/multiple schools (for example schools around Europe) that gather their own resources to create environments where their participants can collaborate. The participants of this VOs contribute their resources to the community or instead pay a fee to support the collaboration (see Figure 1.3). During the collaboration activity, capacities of VO may be required to be expanded. Accounting services within the VO will trigger automatically brokers to allocate resources from external providers. Brokers will select markets to place their bids and allocate the required resources. Many other VOs may have excess of resources. Accounting services then, will trigger seller agents to offer idle resources in a market. This scenario depicts a complete marketplace where participants (VO internal agents) play the role of sellers and buyers, building an overall economy.

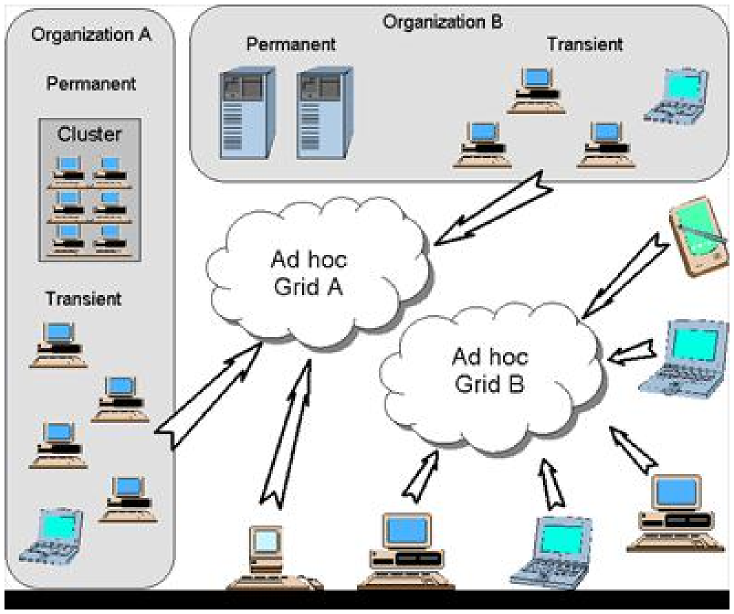


Figure 1.3: Contributory Systems: an example view.

The problem addressed, generically can be defined as a resource management problem in distributed computational environments. Resource management is an issue that has already been studied extensively [1] but the thesis will concentrate on the management and on-demand allocation of resources to VOs from external providers (VOs or Internet users owning idle resources) through the use of auctions. Auctions have been used to regulate the access to resources in a VO but, as far as we know, none effort has been done to enable resource expansion through market institutions from external providers. The use of auctions is motivated by the nature of Contributory Systems (i.e. dynamic and characterized by evolution and heterogeneity) because auctions can adapt to evolution while being simple mechanism able to provide efficient allocations.

The next section introduces the topic of research of this thesis, *resource allocation*, that is one of the most important problems in distributed systems research. Section 1.3 enumerates the main contributions of this thesis while Section 1.4 describes how the dissertation have been structured. Finally Section 1.5 presents the list of publications that made this thesis possible.

1.2 Resource Allocation

Virtual organisations are coordinated groups of individuals and institutions that have common interests and objectives. They share, on the basis of some policies, a set of resources. Participating entities are self-interested but realize the potential benefits from pooling resources. Members may be geographically distributed and can access to the resources any time they are allowed. In this context, we envisage formation and the run-time management of VOs targeting even small organisations. These structures should be able to harness unused networked resources on demand and access computational resources as any other service on the Internet. At creation the VOs should operate with the minimum needed computing resources contributed by the member organisations. At run time resources should be acquired on-demand from resource providers, thus enabling these virtual organisations to perform computational tasks using leased resources. This will extend the notion of utility computing to virtual organisations.

Logically, the organisation that we envisage is that the computational resources on the network are partitioned across virtual organisations. Allocated resources belong to one or other of the virtual organisations whose configuration adapts dynamically to change in load. VOs allocate resources from resource markets when there is a need. This is similar to recent approaches in resource management and provisioning that takes roots from utility computing. Chase et al. [20] present Cluster-On-Demand (COD), a resource management system that allocates servers from a common pool of clustered resources to multiple virtual clusters. Each virtual cluster has independently configured software environments, name spaces, user accounts, and networked storage volumes. Resources are allocated to virtual clusters dynamically on-demand based on sharing policies and adaptive provisioning – the node allotments to virtual clusters change according to competing demands and resource availability.

Shirako [49], a successor of Cluster-On-Demand (COD) extends these principles to resources on a wide-area network, through brokered leasing of resources. Resources are contributed by autonomous sites – resource providers – are pooled and managed by lease brokers. The clients of the leasing service are guest applications – each application is managed by a Service manager that is responsible to negotiate the leases with the brokers on behalf of the guest application.

These architectures show a "two-level" management of resources: each virtual cluster determines internally the mapping of its resources to its applications based on internal objectives and policies. The choice of the internal scheduler and work load management system depends on the type of applications. At the higher level, brokers arbitrate the allocation of global resources to each of these virtual clusters. Virtual cluster respond to load surges by leasing additional resources. It may also react to external resource contentions by accepting reduced service levels and hence free resources.

In the scenario we propose, VOs adapt to changing conditions such as load surges or failures by leasing additional resources. From whom may resources be leased or allocated? Major firms in the computing industry such as IBM, HP, and SUN Microsystems are focusing on agility and flexibility of computing resources and gearing their versions of on-demand computing and IT outsourcing solutions. Utility providers i.e., operators such as Amazon sell raw computing power at fixed prices. Such operators have invested on resources such as cluster farms to furnish utility computing services. But, there are also a large number of idling resources on the Internet. Could we use these resources to offer a utility computing service? Incentives to contribute resources have been driven by public good or collaborative advantage. This is manifested in test-beds such as Seti@HOME [47]. These cannot be generalized since the motivations of contributors are not amenable to generalized models of sharing; so what other incentives for these resource owners and how to arbitrate their allocation? Can a market based approach provide incentives to resource providers?

Economic theory proposed the use of markets and their laws to govern and provide efficient allocation of resources. The MIT Dictionary of Modern Economics [70] defines a market as a *context in which the sale and purchase of goods and services take place.* The Dictionary of economics [77] suggests a definition by which market *is a medium of exchanges between buyers and sellers.* A good is the economic abstraction for a thing that imparts utility to its possessor or recipient. A market transaction takes place when all parties perceive that their own utility will not decrease by their participation, relative to not participating. We consider Tucker's proposal [87], "*a market is a medium in which autonomous agents exchange goods under the guidance of price in order to maximize their own utility*".

Market based resource allocation systems rely on consumers to set a value on the resources that they seek; market rules seek to provide an allocation that is optimal. The fundamental principle is that resources are priced based on the aggregated supply and demand. Consumers are endowed with a budget and seek a quantity of resource that maximizes their internal utility, given the current market price. Trade occurs at a clearing price that balances supply and demand - such allocations are also economically efficient – no reallocation can make one better off without making another worse off. By Adam Smith's invisible hand [83], perfect competition achieves economic efficiency when in a competitive market, in which buyers and sellers act independently and selfishly, channelling scarce resources in an economically efficient way to the users. The invisible hand that guides buyers and sellers is the market price - at this price users buy until their marginal benefit equals price. Pricing of resources becomes the regulatory mechanism to address fluctuations in supply and demand. Price of resources evolve according to market dynamics and indicates a high (high price) or low (low price) demand. This information is used by agents to decide on using the resource or not. Economic market theory [23] states that the prices converge to a stable equilibrium.

You will buy high and sell low.
Brown.
THE INVISIBLE HAND OF THE MARKETPLACE

Figure 1.4: Adam Smith's invisible hand. Image extracted form Institutional advisors web page: http://www.institutionaladvisors.com

Buyya et al. [16] have formulated a set of questions whose answers help to decide if an economy driven resource management system is warranted. We answer some of these questions within the context of our work.

Who are the resource providers and the resource consumers? The majority of the expected resource providers and consumers are domestic owners of computers. Small enterprises and organizations may also provide resources, but more often consume resources. Most of these are now connected to the Internet through broad-band access networks. These consumers do not in general expect free resources, but low cost access to resources. Such consumers seek to reduce their initial IT investment and also running costs. They may be prepared to sacrifice on quality of service, nevertheless do demand an acceptable level of consistency and dependability.

What motivates a provider to contribute their resource to the Grid? Not all VOs search for extra-terrestrial intelligence! Resource owners will contribute resources for a correct compensation. Compensations may be economical, improved QoS, reputation, extended rights or access to services offered by VOs. Generally we expect that most members are more self-interested rather than altruist and expect some benefits in return to the resources that they contribute.

How can users solve there problems within a minimum cost? In general, users can reduce their costs by reducing their investment, both capital and operational expenditure, by considering computational resources as a Utility. Users adjust to market conditions; for example relax dead-lines if this enables them to acquire cheaper resources.

Is access cost the same for peak and off-peak hours? Accessorily what to do when there are more requests than available resources? In the case of excess demand, requests cannot be prioritized – how to give priorities in the Internet? However costs are dissuasive. Basing prices on supply and demand will necessarily motivate rational users to adjust their demand, thus reducing congestion.

How can resource owners maximize profit? They can maximize profit by contributing resources to who value them most, that is, to who pay the most. Users create and form VOs so as to reach a set of objectives. VOs execute applications and services that allow the creators to achieve their goals. The principle issue is that of arbitration when there is excess demand. Straightforward allocation policies are essentially based on priorities perhaps subject to constraints on quota of utilisation. If self-interested participants are free to set their own priorities then they will each specify the maximum priority since they do not have the incentive to do the contrary. Moreover this approach does not provide incentives to owners to share their resources unless of course the each provider is also a consumer.

Thus, at a higher level, when inter-VO resource management is addressed, market-based resource management is expected to be the most satisfactory when participants are dominantly self-interested. Resource providers or owners have incentives to share their resources if they are adequately compensated. Pricing of resources establishes a common scale of value across various resources and resources are allocated to those who value them the most.

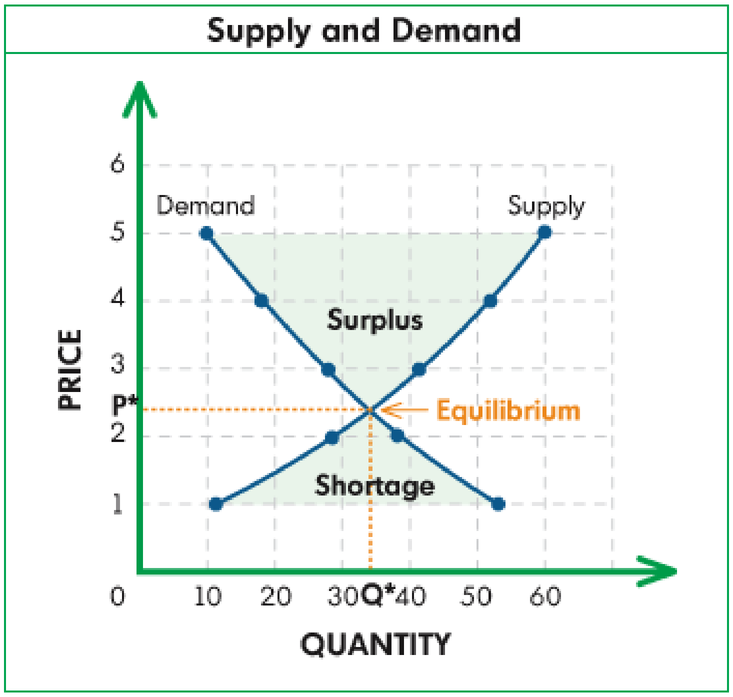


Figure 1.5: Supply and demand curves and equilibrium point. Image extracted from the economic blog: http://enthusiasm.cozy.org/

1.3 Contributions

The major contributions of this thesis are the following:

1. The semantic analysis of computational resources and the subsequent design and implementation of a formal bidding specification and language specially adapted to computational resources. The presented bidding specification provides and expressive and Grid-oriented bidding language that can be used in different auction institutions.

2. The design and the implementation of DyMRA[2] - A reliable peer-to-peer based resource allocation framework that has been specially designed for Contributory Systems. DyMRA provides the main components to enable dynamic resource expansion in Contributory Systems having been specially designed to cope with dynamism and service availability.

3. The design and the implementation of CAS[3] - A generic market mechanism holder component that provides functionalities to support, manage and configure different auction formats. CAS enables the cohabitation of auctions in a marketplace which makes our approach one step further from current existing frameworks.

4. The design, the implementation and evaluation of MLDA[4] - An auction mechanism adapted to trade time-differentiated items that until now were traded in multiple auction instances.

[2]Dynamic Market Deployment for Resource Allocation
[3]Configurable Auction Server
[4]Multi-Lane Double Auction

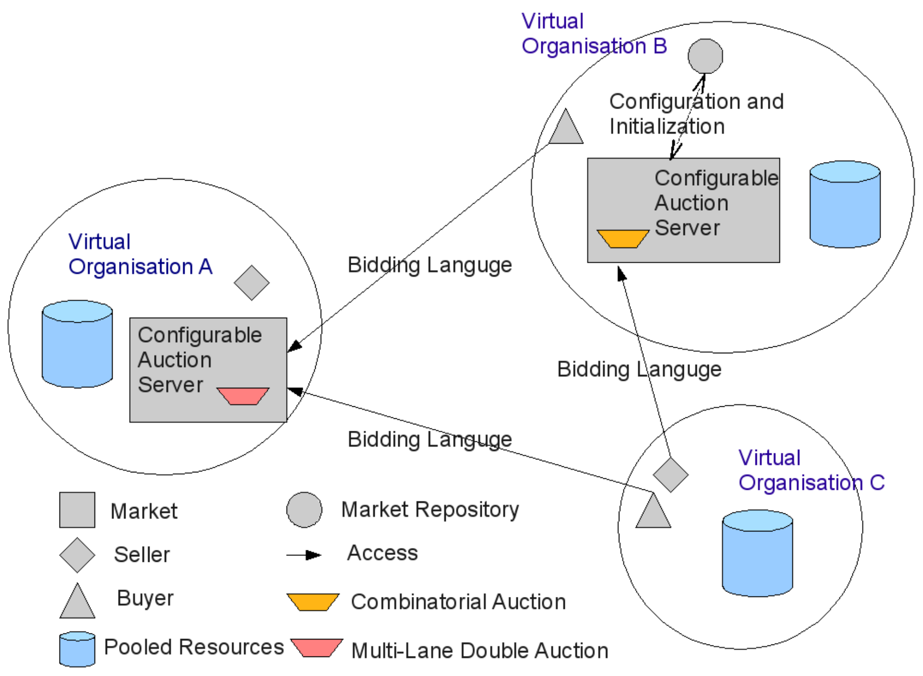


Figure 1.6: Target scenario of the thesis. Overall scenario where different VO share/offer their resources to be leased by others. Requirements are expressed through a well-defined bidding specification while mediation and allocation is carried out through market mechanisms. Markets are configurable entities than run in the scope of a VO and expose their functionalities through well defined APIs. Brokering services are used to access discover markets and participate in them while the overall logic is transparent to the end-user.

Figure 1.6 presents the targeted scenario. Internal logic in Virtual Organizations will automatically trigger resource allocation from external providers when internal resources become scarce. The first contribution of the thesis proposes a bidding specification that will be used to acquire/bid in markets. Both sellers and buyers will make use of the bidding specification that will be especially designed for computational resources. Buyer agents and seller agents will be autonomous entities running in the scope of a VO. Internal policies and strategies will trigger them so as to obtain or offer resources from/to others.

Resources will be offered through markets that will run in a generic container exposed by the selling VO (alternatively, other VOs can host the component). The container will offer functionalities to be configured and to support different auction formats. Auctions will be the main market mechanisms used to mediate the allocation of resources. The last contribution of the thesis is an auction specially adapted to trade time-differentiated resources as Grid resources are.

1.4 Outline

The material in this thesis is structured as follows:

- **Chapter 2** discusses first the state of the art relevant to the work in this thesis. The chapter characterizes the scenario where the work of this thesis is carried out. It presents the related approaches of systems where resources are provided by their participants to achieve a common objective. Subsequently, the state of the art of market based resource allocation frameworks is presented. The section discusses issues not already addressed in current existing approaches and motivates the contributions of the thesis.

- **Chapter 3** presents the first contribution of this thesis, namely it presents an approach for resource specification and description. It includes an analysis of computational resources and a semantic description of their properties. The semantic description provides the basis for a formal bidding specification that enables bid and offer formulation for computational marketplaces. Finally, Chapter 3 presents the implementation of the bidding language and compares it with related work.

- **Chapter 4** proposes an architecture to manage resource expansion and offering in environments where resources are provided by their participants, ad-hoc communities and Open Grids, the second contribution of the thesis. The architecture is based in the peer-to-peer paradigm and it has been specially designed for systems subject to high levels of dynamism, heterogeneity and constant evolution. Resource allocation is handled by market mechanisms (i.e. auctions). The architecture proposes a generic auction component able to be configured and adapted to different type of mechanisms in order to handle diversity of resources and diversity of demand. The chapter also discusses the benefits of the cohabitation of multiple market mechanism in a Grid.

- **Chapter 5** focuses in the most economics-based part of the thesis and presents an approach to develop a market mechanism specially adapted to trade computational resources. The chapter introduces the market mechanism spectrum and presents the third contribution of the thesis. The chapter presents a novel auction that extends current existing approaches by enabling the allocation of time-differentiated resources by the same market instance. This type of auction is specially adapted to computational environments such as the Grid where resources are leased rather than acquired.
- **Chpater 6** presents our conclusions, future directions and some open research questions.

1.5 Publication Record

Portions of work presented in this thesis have been partially or completely derived from the following set of publications. Before each publication in brackets the percentage of our contribution is presented. The percentage of contribution indicates the amount of work, ownership and responsibility when doing the research related to the publication.

Chapter 3 is partially derived from the following publications.

1. (20%) Konstantinos Kotis, George A. Vouros, Alexandros Valarakos, Andreas Papasalouros, Xavier Vilajosana, Ruby Krishnaswamy, Nejla Amara-Hachmi , The Grid4All ontology for the retrieval of traded resources in a market-oriented environment. International Journal of Web and Grid Services (IJWGS) ISSN (Online): 1741-1114 - ISSN (Print): 1741-1106, (2009).
2. (90%) Vilajosana, X.; Marquès, J.; Krishnaswamy, R.; Juan, A. (2009): A Bidding Specification for Grid Resources. Int. Journal of Grid and Utility Computing. ISSN:

1741-847X

3. (90%) Vilajosana, X.; Marquès, J.; Krishnaswamy, R.; Juan, A.; Amara, N.; Navarro, L. (2008): Bidding support for computational resources. In Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems. Barcelona, Spain, March 4-7. p309 - 315.ISBN: 0-7695-3109-1.

4. (20%) Konstantinos Kotis, George A. Vouros, Alexandros Valarakos, Andreas Papasalouros, Xavier Vilajosana, Ruby Krishnaswamy, Nejla Amara-Hachmi. (2008) The Grid4All ontology for the retrieval of traded resources in a market-oriented Grid. In Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems. Barcelona, Spain, March 4-7. ISBN: 0-7695-3109-1

Chapter 4 is partially derived from the following publications.

1. (40%) Nejla Amara-Hachmi, Xavier Vilajosana, Ruby Krishnaswamy, Leandro Navarro-Moldes, Joan Manuel Marquès: Towards an Open Grid Marketplace Framework for Resources Trade. Lecture Notes in Computer Sciences.OTM Conferences (2) 2007: Vilamoura (Portugal).1322-1330.

2. (40%) Daniel Lázaro, Xavier Vilajosana, Joan Manuel Marquès: DyMRA: Dynamic Market Deployment for Decentralized Resource Allocation. Lecture Notes in Computer Sciences. OTM Workshops (1) 2007:Vilamoura (Portugal). 53-63.

3. (25%) Xavier León, Xavier Vilajosana, Rene Brunner, Ruby Krishnaswamy, Leandro Navarro,Felix Freitag, Joan Manuel Marquès; Information and regulation in decentralized marketplaces for P2P-Grids. In the Proceedings of Fourth Collaborative P2P systems (COPS08) Workshop, Rome, Italy, June 23-25.

4. (60%) Vilajosana, X., Lázaro, D., Marquès, J., Juan, A. (2008): Towards decentralized resource allocation for collaborative peer to peer learning environments. In Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems. Barcelona, Spain, March 4-7. p. 501-507. ISBN: 0-7695-3109-1

5. (45%) Daniel Lázaro, Xavier Vilajosana, Joan Manuel Marquès, Angel A. Juan, A Framework for Dynamic Resource Allocation in Decentralized Environments, International Transactions on Systems Science and Applications Journal. ISSN 1751-1461.

6. (75%) Vilajosana, X., Lázaro, D., Marquès, J.M., Juan, A. (2009). DyMRA: A decentralized resource allocation framework for collaborative learning environments. Book: Intelligent Collaborative e-Learning Systems and Applications. Series "Studies in Computational Intelligence". Berlin, Germany: Springer-Verlag. (*To appear.*)

7. (80%) Vilajosana, X., Krishnaswamy, R., Marquès, J.M. (2009). A Configurable Auction Server for Resource Allocation in Grid. In Proceedings of the Third International Conference on Complex, Intelligent and Software Intensive Systems. Fukuoka, Japan, 16-19 March 2009.

The scope of the research is motivated by the following publication:

1. (25%) Joan Manuel Marquès, Xavier Vilajosana, Thanasis Daradoumis, Leandro Navarro: LaCOLLA: Middleware for Self-Sufficient Online Collaboration. IEEE Internet Computing 11(2): 56-64 (2007).

2. (20%) Ruby Krishnaswamy, Leandro Navarro, René Brunner, Xavier León, Xavier Vilajosana: Grid4All: Open Market Places for Democratic Grids. Lecture Notes in Computer Sciences. GECON 2008: ISBN: 978-3-540-85484-5. 197-207

Chapter 2

Resource Management and Allocation

2.1 Sate of the art in Contributory Systems

This section presents the state of the art of current existing systems where resources are contributed by their participants. The section identifies a collection of existing systems where the research carried out in this thesis can be applied. Thus, systems described below constitute perfect examples of the scenarios that we are addressing in this dissertation.

2.1.1 Collaborative Grids

Collaborative Grids consists of several participants which agree to aggregate their resources for a common goal. The OurGrid project [7] is a typical example of such systems. It proposes mechanisms for research centres to put together their local Grids. A mechanism allows the local resource managers to construct a P2P network. These solutions are attractive because utilization of computing power by scientists is usually not constant. When scientists need an extra computing power, this setup allows them to access easily their friend universities resources. In exchange, when their resources are idle, it can be given or rented to others' university. This requires a cooperation of the local VO, usually at the resources managers' level, and mechanisms to schedule several applications.

A similar approach has been proposed by the Condor team under the term "flock of Condor" [14]. OurGrid has been in production since December 2004 and today aggregates computing resources from about 180 nodes shared by 12 peers. The platform has been limited to supporting bag-of-tasks. Local users have always the priority for their tasks on their local resources, only the unused local resources are shared with other peers. Local jobs kill remote jobs if needed. For promoting cooperation amongst peers and avoid freeriders [2], OurGrid use a network of favours. Each peer maintains a matrix of the computing time that it gets granted from other peers. Then, if a processor is requested by more than one peer, it allocates it to the peer with the greatest favour. The favour computation is protected against malicious peers that, for example, would reset its state in order to gain more computing time. Other peers are discovered through a centralized discovery system. The network is a free-to-join Grid, where remote peers are not trusted. To address this issue, a sandbox mechanism is proposed (Sand-boxing Without A Name). Several resource allocation policies have been experimented. The first one, Workqueue with Replication (WQR), was simply sending a random task to the first free processor found. In version 2.0 of OurGrid, a new scheduler tries to avoid communication cost by introducing storage affinity. Tasks are sent to computing nodes that are closest to used data. This algorithm tries to avoid the need for redundant information about tasks such as expected completion time. The first algorithm was found to be still more efficient on some cpu-intensive workloads.

LaCOLLA [56, 60] is a peer-to-peer grid optimized for collaborative interaction and resource sharing. LaCOLLA middleware provides collaborative functionalities to let participants in collaborative activities self-organize using only the resources that themselves provide. By nature LaCOLLA can be considered a Contributory System based on decentralized components with autonomous behaviour. It uses algorithms based on epidemic propagation of information. When two sites communicate exchange their local information as well as third party information received in previous interactions. Besides, LaCOLLA relies on optimistic replication techniques to ensure awareness and availability of the information in a group in collaboration. Randomization is exploited as a technique to improve failure resilience and scalability.

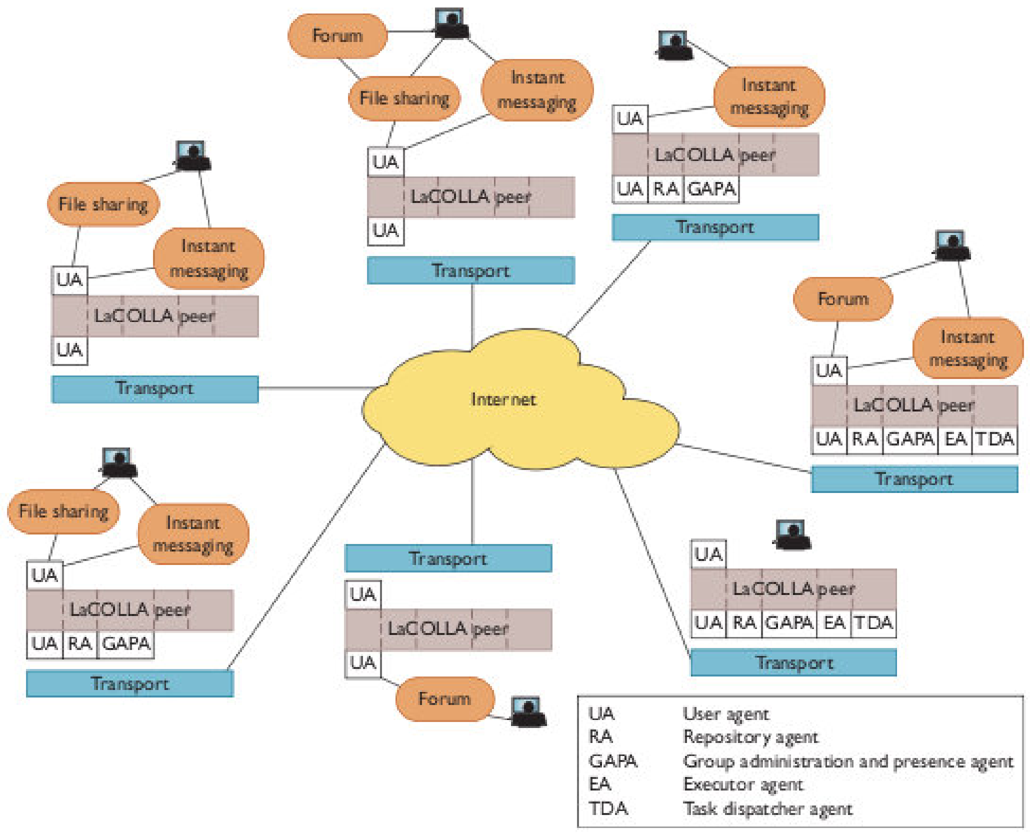


Figure 2.1: Architecture of LaCOLLA.

In LaCOLLA, participants provide the computational resources (storage and processing) and applications that users need to carry out the groups activities. LaCOLLA manages them in such a way that any resource owner can disconnect them at any time without warning and without affecting the groups functionality. The middleware then guarantees, despite intermittent node and network availability, that the system can self-organize without requiring the participants to manually intervene. In the end, the collectivism exists if the participants ensure the groups self-sufficiency by providing sufficient resources.

LaCOLLA has been designed to maintain group members' freedom; members can decide to disconnect resources from the group at any moment. While a resource is within the group, however, the group is in charge of its administration and the group policies apply. Resources provided to a group are used transparently that is, users don't know which resources they're using to carry out an action. Thus, members carry out their actions with group resources in line with group policies. Contributing resources for the benefit of the group makes sense in environments in which participants share more than just resources, values, or a common goal (for example, groups within a company, student projects, research groups with researchers from different organizations, and non- governmental organizations [NGOs]). The architecture of LaCOLLA can be seen in Figure 2.1. LaCOLLA middleware is implemented in JAVA and available at [46]. Current implementation of LaCOLLA manages VO resources following group specific policies, however until now LACOLLA is not able to expand VO capacity by allocating external resources, even authors considered this as a future work.

2.1.2 Internet Volunteer Grids

Internet Volunteer Grids systems have been amongst the largest distributed systems in the world. Projects such as SETI@Home or distributed.net are able to provide hundreds of TFlops on dedicated application from hundred of thousands nodes. For over a decade, the largest distributed computing platforms in the world have been Internet Volunteer Grids, which use the idle computing power and free storage of a large set of networked (and often shared) hosts to support large-scale applications. In this case of Grid, owners of resources are end-user Internet volunteer who provide their personal computer for free. Internet Volunteer Grids are an extremely attractive platform because there offer huge computational power at relatively low cost. Currently, many projects, such as SETI@home [47], FOLDING@home [35], and EINSTEIN@home [4], use TeraFlops of computing power of hundreds of thousands of desktop PCs to execute large, high-throughput applications from a variety of scientific domains, including computational biology, astronomy, and physics.

The Great Internet Mersenne Prime Search (GIMPS) [48] is one of the oldest computation using resources contributed by their users. Its started in 1996 and is still running. Each client connects to a central server (PrimeNet) to get some jobs. Resources are divided in 3 class based on the processor model and gets different type of tasks. The program only uses few computational resources of the client and does very little communications with the server (permanent connection is not required). The program checkpoints every half hour.

Since 1997, Distributed.net [29] tries to solve cryptographic challenges. RC5 and several DES challenges have been solved. The first version of SETI@Home [47] has been released in may 1999. There was already 400 000 pre-registered volunteers. 200 000 clients registered the first week. Between July 2001 and July 2002, the platform computed 221.10 workunits at an average rate of 27.36 TeraFLOPS. The programs is doing some treatments on a signal recorded by a radio-telescope and then search for particular artificially made signal in it. The original record is split in workunit both by time (107s long) and by frequency (10 KHz).

The Electric Sheep screen-saver [30] "realizes the collective dream of sleeping computers". It harnesses the power of idle computers (because they are running the screen-saver) to render, using a genetic algorithm, the fractal animation displayed by itself. The computation uses the volunteers to decide which animation is beautiful and should be improved. This system consists only of one application but, as the project web site claims, about 30000 unique IP addresses contact the server each day and 2Tb are transferred.

All these application projects share many common components. So, there was a need for a platform that would provide all these components. Only the part that really does the computation need to be changed for each project. One proposed platform is The Berkeley Open Infrastructure for Network Computing (BOINC) [6] that is the biggest volunteer computing platform. More than 900 000 users from nearly all countries participate with more than 1 300 000 computers. More than 40 projects, not including private projects, are available including the popular SETI@Home project.

Each client (computing node) is manually attached by the user to one or more projects (servers). There is no central server and most of the scheduling is done by clients. The server is made of several daemons. A daemon that takes care of the different states of the workunit life cycle, replicates (redundancy) the workunit in several results (instances of workunits). Each result is executed on a different client. Then, back to the server, each result is checked by the validator before being stored in the project science database by the assimilator. All communications are done using CGI programs on the project server, so, only port 80 and client to server connections are needed. Each user is rewarded with credits, or virtual currency, for the count of cycles used on its computer.

The POPCORN [76] is a platform for global distributed computing over the Internet. It has been available from mid 1997 till mid 1998. Today, only the papers and documentation are still available. This platform runs on the Java platform and tasks are executed on workers as "computelets", a system similar to usual Java applets. Computelets need only to be instantiated for a task to be distributed. Error and verification process is left to the application level. The platform provides a debugging tool that shows the tree of spawned computelets (for debugging concurrency issues). There is also a market system that enables users to sell their CPU time. The currency works almost the same as BOINC credits. Some applications have been tested on the platform (brute force breaking, genetic algorithm...).

Bayanihan [80] is another platform for volunteer computing over the Internet. It is written in Java and uses Hord, a package similar to Suns RMI for communications. Many clients (applet started from a web browser or command line applications) connect to one or more servers.

This section presented example applications and middleware approaches built following the concept of contribution. These applications mainly propose to take advantage of idle resources on the extremes of Internet which we also consider a new opportunity to provide resource sharing and expansion functionalities in Contributory Systems. Several lessons have been learnt from the presented applications, the need for regulation to access resources and to avoid freeriding and the enormous capacities that can be obtained from the nodes at the extremes of the Internet. Besides, the work presented in this thesis addresses scenarios characterized by groups of participants that provide their own resources to meet a common objective. Collaborative Grids are a subset of Contributory Systems where the work of this thesis can be applied. Volunteer Grids can also take advantage of the work proposed in this dissertation since Volunteer Grid capacity can be expanded by means of allocation of resources from external providers.

2.2 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 organization that helps to shape outcomes. Intuitively [68] a mechanism solves a problem by assuring that the required allocation occurs when agents choose their strategies to maximize 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 sections gathers the most significant auction based resource allocation frameworks found in the literature.

2.2.1 Auction markets for single type of resource

SPAWN [89] was designed to tap in 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 generalized 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. [73] present a trading platform for storage services. The platform implements a centralized 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 trading of a single type of homogeneous resources. These have the benefit of reducing communication costs (single bids), and with suitable pricing policies are also incentive compatible.

2.2.2 Auction markets for multiple types of resources

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. [21] present a resource discovery and allocation system where users may express preferences using a bidding language supporting XOR bids, at most one of the preferences is to be allocated. Multiple resources may be requested to a central auction server that clears periodically. Resource requests are for fixed durations of times 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. [82] present an iterative combinatorial auction that maximizes 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 optimizing the revenue. Prices are calculated using the approach presented within [55] - prices are the dual solution to the primal Linear Programming Problem (LP).

Schnizler and Neumann [81] present a multi-attribute combinatorial auction that maximizes 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 [9] 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 [3]. 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 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 virtual currency. Virtual currency is the amount of credit a site has, which is directly determined by the sites overall resource contribution to the federated system. Bellagio employs Share [22] 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 charging 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.

Proportional sharing markets for divisible resources

Market based proportional sharing models are one of the most popular approach in problem-solving environments. Basically this approach consists of allocating to 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 [34] is a system designed for time-sharing networked nodes such as in Planet-Lab; 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 [51] auction where 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 – 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 maximize their utility. This system is intrinsically decentralized as the maximization of utility is done locally at each user. This system is appropriate for divisible usage of CPU, an assumption that may not be acceptable to a wide range of applications.

Decentralized markets for single type of resources

Peer-to-peer auctioning has emerged as a new computing paradigm to decentralize auction processes. Many systems address the decentralized 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 [17, 40, 75, 84].

PeerMart [43] distributes brokering in an auction based allocation mechanism. Auctioneers rather than being a single broker are formed by a set of peers which synchronize to clear a double auction market. Consumers and providers are distributed in the Pastry overlay network [17]. Broker sets are formed by some nodes in the overlay. The double auction they implement clears continuously. For each allocation, brokers synchronize their information in order to determine who the winners are and to avoid malicious peers. Synchronization is carried out through 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. [86] make use of CAN [75] to build a market architecture. They propose to distribute peers in different sub-regions and assign 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 responsible broker 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 to replicate 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 of Despotovic et.al. [28], 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. [32] 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 [59]. 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 centralization 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 [8] 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 a costly. Decentralized auctions usually map one item to one responsible broker. In some other approaches, auction decentralization can be compared to a distributed sorting problems and solved according to that. 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 decentralization can only come when multiple items are traded assigning different responsible brokers for each item. As will be seen in Chapter 4, in our work decentralization will be attained by short-lived market instances since we consider co-allocation and substitute allocation important aspects that need to be attained.

2.3 Related work on architecture of computational resource markets

Most relevant architectures to the work carried out in this thesis are the GRACE [16], OCEAN [69] and CATNETS [18, 33] projects. GRACE is one of the first systems to propose a comprehensive architecture to promote the Economy-based Grid. It has established some requisites for any economy-based Grid system, such as publishing and discovering of Grid entities, resource pricing schemes, economic models and negotiation protocols, etc. OCEAN presents an open software infrastructure to automate trading of heterogeneous computing resources on the Internet. In contrast to GRACE, this infrastructure aims a completely decentralized architecture facilitated by a peer-to-peer search protocol that can quickly find suitable matches among large number of providers and consumers. The multi-layered OCEAN infrastructure is architected to be portable over various cluster and Grid infrastructures. CATNETS has a middleware architecture that purports to provide economic and market based resource management systems.

CATNETS is a completely decentralized architecture where bilateral negotiations are conducted between providers and consumers. It aims to remove the shortcoming of centralized co-ordination by allowing peers to continue to obtain services even if there are failures and withdrawal. The layered software architecture of CATNETS isolates economic agents from the underlying Application Layer Network by providing support for pluggable mechanisms and strategies. The decentralized peer-to-peer architectures of OCEAN and CATNETS based on bilateral negotiations make bundled allocation a complex process. Often Grid applications mandate multiple types of resource.

The GRACE architecture is more adapted to Grid systems formed of a small number of nodes - where each node represents a large clustered system. OCEAN lacks support for the implementation of multiple auction mechanisms, and in fact does not address this need.

Sharp [38] is a framework for secure distributed resource management. Participant sites can trade their resources with peering partners or contribute them to a peer federation according to the local site sharing policies. SHARP framework relies on the bartering economy as the basis to exchange resources between resource domains. A cryptographically signed object called Resource Tickets (RTs) is issued by each participating site. These RTs are exchanged between the participating sites for facilitating coordinated resource management. In Sharp framework, every participating site is completely autonomous and holds all the rights over the local resources. Sharp framework considers collection of logical sites or domains, each running local schedulers for physical resources (e.g. processors, memory, storage, network links, sensors) under its control. The fundamental resource management software entities in Sharp include site authority, service manager and agents. These entities connect to each other based on a peer-to-peer network model. The resources at each site are managed by a site authority, which maintains the hard state. The site authority accepts the resource claims presented by remote sites in the system. Resource agents mediate between site authorities and resource consumers (service managers). A resource is allocated to the service manager at a SHARP site using the two-phase negotiation process. Initially, a service manager obtains a resource claim in the form of a ticket from an agent. In the second phase, the service manager presents the ticket to the appropriate site authority to redeem it. The authority may reject the ticket or it may honour the request by issuing a lease for any subset of resources or terms specified in the ticket. A lease is hard claim over concrete resources, and is guaranteed valid for its term unless a failure occurs.

Systems such as PeerMart [43], described earlier in this document, implement distributed auctions by associating a broker for each resource/service type. Brokers are implemented as peer-sets on a structured P2P overlay. In PeerMart segmentation of markets puts the onus on the clients to choose between equivalent resources. PeerMart auctions are neither generic nor configurable and authors do not address these issue.

GridBus [15] uses non-coordinated brokers to match tasks to resources. Matchings are provided by a single mechanism that allocates asks to bids based on some QoS specifications. This mechanism is not optimal but configurable and constitutes a first approach to enabling adaptability. The system is organized as a federation of resource nodes managed by a set of co-operating brokers. Brokers of each node either locally accept a submitted job or forward it to the cheapest remote node. The assumption of co-operation is in general not warranted.

Recently, the project SORMA [66] which deals with methods and tools for efficient resource allocation, through a self-organizing resource management system, using market-driven models supported by extensions to Grid infrastructures which is a parallel approach to the work carried out in this thesis.

2.4 Lessons Learnt

Table 2.1 presents a summary of the most important architectural approaches to resource allocation in large scale computational platforms. The table presents how negotiation is carried out in those systems, being market-based approaches the most used mechanisms to manage resource allocations. In this thesis auctions will be used as mechanisms to allocate resources because they have demonstrated that can provide efficient allocations by only requiring user's to express their utility as a single value (i.e. price). The table also presents how negotiation is structured, that is, either by single responsible components or instead by decentralized services. It is clear that decentralization improves failure resilience and scalability which is one important aspect to be considered when designing large scale systems. In contrast to all presented approaches, the thesis proposes to make use of on-demand created markets in order to provide this functionality when needed. Contributory systems are characterized by dynamism and evolution which require on the one hand to do not overload the system with unnecessary services and on the other hand to adapt mechanisms to the currently required/offered resources. How matchings are provided is an important aspect that is dealt in different ways by the presented approaches. Central marketplaces or distributed brokers implementing an auction are the main models used. Besides some of those systems point out the need to support multiple cohabiting market institutions. Our thesis proposes the cohabitation of multiple market mechanisms held by a common configurable framework.

.

	Negotiation	Matching	Multiple Markets
OCEAN	Central Marketplace	Search algorithm	Not addressed
CATNETS	Bilateral negotiation	Exchanges	Demonstrate the need
GRACE	Central Marketplace	Multiple market mechanisms	Demonstrate the need
SHARP	Coordinated resource sharing	Token-based exchanges	Not addressed
SORMA	Central marketplace	Multiple market mechanisms (Auctions)	Demonstrate the need
PeeMart	Distributed brokers	Market Mechanism (Double Auction)	Not addressed
GRIDBUS	Distributed brokers	QoS based scheduling	Demonstrate the need

Table 2.1: Summary of existing approaches and their approach to resource allocation

The important lesson learnt from these architectures is the requirement for open APIs and layered software architecture. An open market place for Contributory Systems needs to address both in terms of development and run-time co-habitation of multiple market mechanisms. Even though CATNETS, SORMA and GRACE demonstrate the need to provide support for multiple market mechanisms, the aspect of inter-operability of agents in the face of multiple market negotiation protocols is not addressed. The increasing spectrum of Collaborative and Volunteer computing applications moving computation to the extremes of Internet, make inter-VO resource allocation a promising opportunity to democratize access to computational resources. New utility computing oriented infrastructures will appear and resource management will constitute the key point for its feasibility. The architectures presented in this chapter, even decentralized, still do not provide flexibility and configurability to support dynamic and evolving environments such as the ones characterized by Contributory Systems. One consequence of open market places and not addressed until now, is the need to provide support for both designers of market protocols and for participants at the market place. This will be realized through flexible and configurable architectures and semantic description of resources. We have taken the approach to define the markets in a structured way; this is a requirement to match mechanisms to application scenarios and to provide methodologies and tools to designers of market negotiation protocols. Next chapter presents the first important contribution of this thesis. A semantic description of resources and a generic and well defined bidding specification are presented as a tools to enable flexible and adaptable bidding when multiple types of markets co-habitate.

Chapter 3

Computational Resources

3.1 Introduction

This chapter aims to study in deep the properties of the resources in computational environments such as an Open Grids and a Contributory System. In the context of this thesis, a comprehensive understanding of the nature of computational resources is of relevance and fundamental to develop matchmaking and bidding specification languages. The nature of computational resources in such a distributed and dynamic environment has to be understood so as to facilitate its description. Description of resources is also fundamental in environments where users need to describe their requirements in order to fulfil their objectives. Open Grids are characterized by the dynamism and heterogeneity of their resources, users and applications. Such heterogeneity, constitutes a trade off for any specification language since it has to enable precise description of resources but also be flexibile in order to be extended when the nature of resources evolve. This chapter describes the main properties of Grid resources, the described properties can be considered inherent to resources but those properties that derive from the resource allocation process are also pointed out.

In this chapter we present a semantic approach to resource description, bid and offer configuration in a scenario where the allocation of resources is guided by market mechanisms like auctions. In this context, semantics are used to build a common base of knowledge that is used by buyers and sellers in the computational marketplace so as to enable the matchmaking of resources. Thus, an ontology based approach is presented to relate and describe the main concepts regarding resources in a computational environment. Besides, the chapter presents an bidding specification language that has specially designed for economical based resource allocation systems. The language has been specified taking in consideration the concepts defined by the ontology. An implementation is also provided as an XML schema [27]. The key objective of the bidding specification is to become a generic language that can be used by any market mechanism. To fulfil that requirement, a set of tools have been developed in order to isolate the specification of the bid/ask to the specific market implementation.

Next section introduces the main properties that the thesis addresses regarding computational resources. Section 3.3 motivates the use of semantics to build a common base of knowledge so that bidders bids and sellers asks can be matched. Section 3.4 presents the main concepts of the developed ontology. Section 3.5 introduces the need for a common bidding specification based on the previous section. In section 3.6 the motivation for a formal bidding language is presented, section 3.7 presents the related work on bidding specification and languages and outlines the main advantages of our approach. Next section, presents the design and implementation aspects of the language. Section 3.9 improves the bidding specification by introducing concepts to relate bids. Section 3.10 presents a set of tools to decompose bids making them suitable for any auction format, section 3.11 evaluates the decomposition tools introduced in the previous section and finally section 3.12 concludes the chapter with an outlook to future work.

3.2 Resource Properties

This section aims to identify the main properties of Grid resources for a correct bidding language design.

1. **Divisibility and Shareability:** Grid resources are mainly continuous resources

and are typically discretised in some dimension by dividing it into a set of indivisible units. Processing capacity may be traded by dividing in time-units where each time-unit is traded to a single consumer, or multiple consumers may be allowed to share the capacity; the latter is more often the case of network bandwidth where different flows are multiplexed, but with guarantees of requested bandwidth being satisfied over a period of time. Hence division of continuous resources may occur both in time and space. In this context, the consumers (buyers) and providers (sellers) should be able to configure their offers and bids in a way that best suits them.

2. **Single Items or Multiple Items:** A single item refers to a resource that is traded as one atomic item. However the notion of what an 'item' is should be configurable at the market. One seller may want to trade bundles of 4 CPUs for 4 hours, as one indivisible set and another 2 CPUs. Secondly the notion of 'composite' resources are relevant in Grid settings: in a simple case, it does not make sense to trade CPU and volatile memory as separate resources and in more complex cases, providers (or aggregating resellers), may want to sell units of composite resources. For example, aggregated processing, storage capacities, bundled with a set of applications and middleware.

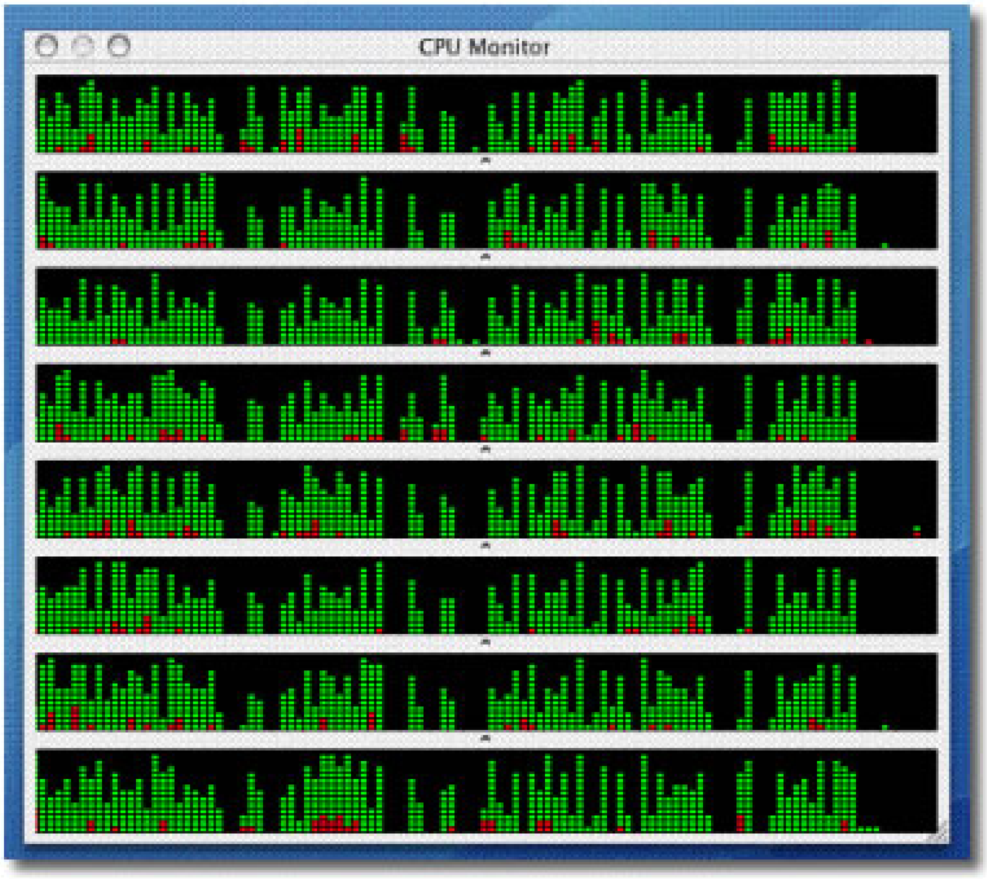


Figure 3.1: Divisibility of resources.

3. **Single Unit or Multiple Unit:** Markets may trade in single or multiple units of an item. In the case of resource markets, it is more often the case where multiple units of anonymous and indistinguishable items are traded, such as multiple units of CPUs for multiple time-slots. Bidding support needs to provide a flexible and compact way to represent the quantity preferences for both buyers and sellers.

4. **Time Factor:** Grid resources are leased. Consumers may have constraints on when the resources are needed and the duration of allocation; similarly providers may trade their resources only for specific times. Hence the bid (and offer) must be able

to specify the time ranges within which resources are required and their duration. A typical bid that needs to be supported is that of a consumer that requires two CPUs satisfying attribute description $= CPU > 1\ GHz,\ mem > 1GB,\ disk > 20GB$ for 10 time slots between 10:00 and 18:00 assuming that time-slots are defined to be of 30 minutes. Within single item auctions, the time-slot may also be considered as the item.

5. **Pricing:** Linear pricing sets the same price for each item of resource; bundle pricing sets the price for the entire bundle. Bundle pricing increases the complexity since allocation must choose those bundles that maximize the objective function.

3.3 Matchmaking and description of resources

Trading resources based on the demand/supply of users is the main driver in the competitive, economic market model: In the context of a Contributory Systems environment, VOs willing to offer their resources are in competition with other VOs, and, of course, computational resource owners are in competition with Grid service providers: In a market-oriented environment computational resources and services are made available through markets initiated either by resource providers, participants of the VO with idle resources, resource consumers or third party entities (usually brokering agents within the VO). Markets are initiated by means of resource orders that peers issue in a distributed manner. Such an order can be either the request of a consumer, or an offer of a provider. Adopting such a distributed market model, resource consumers and providers negotiate over resources using auctions that run in markets. Given the distributed nature of market creation and management, in order for an auction to take place, and thus resources to be allocated to peers, markets must be understood by potential participants. Next section presents an ontology (see description below) that represents types of resources being offered and requested by peers, offers and requests that can be made by peers, types of markets where orders are placed, as well as market related properties.

3.4 Ontologies for resource description

Computational resources are heterogeneous and geographically distributed with varying availability and variety of usage for diverse users at different times. In order to enable trade of these resources (computational resources, storage resources) between providers and consumers, they need to be modelled using a formalism that allows their semantic description, for their publishing, storing, and discovery. A semantic description is required so as to provide flexibility to any descriptive language because described entities would be related by their constituting properties. Besides, the required resource description should support and ensure a common understanding (about resources availability, location, performance, economic characteristics, etc.) among the trade participants as well as the different services (such as resource advertising, retrieving, etc.) contributing to their interoperability.

To enable participants in a trading to share a semantic description of grid resources via a formal language an ontology-based approach seems to be suitable. An ontology [42] is a formal explicit description of concepts in a domain of discourse (concepts), properties of each concept describing various features and attributes of the concept (roles or properties), and restrictions on slots (role restrictions). Ontology is used in order to:

- Share common understanding of the structure of information among people or software agents.
- Enable reuse of domain knowledge.
- Make domain assumptions explicit.
- Separate domain knowledge from the operational knowledge.
- Analyse domain knowledge.

3.4.1 How ontologies meet our requirements?

The presented resources ontology aims at the description at an abstract level- of the traded resources (physical resources, computational resources, etc.). This ontology has to sketch out the principal classes and properties of the traded grid resources.

Resources are traded by applying economic models: They are traded as goods, in markets, based on the supply and demand law. An explicit and commonly agreed formal description of resources and their exchange (trading) among information consumers and resource providers is needed in highly distributed and heterogeneous Grid environments.

This shared description should support and ensure a common understanding (communication of knowledge about resources availability and needs) of members (resource consumers and providers) within a market.

3.4.2 Representation of traded resources

As already pointed, the representation concerns a) the types and characteristics of the resources, b) properties/constraints related to the specific offers and requests, c) properties of the markets to which the specific orders are placed.

Resources

The proposed ontology represents knowledge about the different types of resources available in a Ad-hoc Grid computing environment (also in a Grid environment). A resource in the context of Grid computing may be defined as any passive entity required for implementing functionality. Resources are entities that are offered by resource providers or requested by resource consumers, and that are traded in e-markets initiated by consumers, providers and third- parties. In addition to resources, applications or services may also be needed to become available through markets: However we do not deal with services and applications in this dissertation.

In the proposed approach we mainly deal with the representation of computational and storage resources, since these are widely accepted as a first-priority when addressing Grid resource markets. Resources, such as computational and storage resources are dealt at a logical rather than at a physical level. This choice is driven by the fact that ordinary

Internet users are less concerned about the specific hardware properties of a resource and more about the service rendered by the resource. Taking the example of hard (persistent) storage, the technology used to implement the persistent storage (whether be Flash, magnetic disks) and the controller technology (whether be SCSI, IDE) is irrelevant to the user, since this is hidden by software which virtualises the underlying technologies and provides a common higher level user interface to store and retrieve storage. The user is nevertheless concerned about the quality, quantity, and performance of the hard disk (such as throughput, error rates) which are commonly-agreed and widely-known metrics.

More specifically, the following holds for the representation of resources:

- Any Resource has a specific Identifier that is related to a specific Service via which it is exposed to any (human or software) Agent.
- A Hardware Resource has a capacity property specifying its capacity with respect to a specific unit of measurement.
- A resource is hosted by a Machine. A machine can host one or more resources, given that they are all Atomic resources (they are not Composite or Aggregated resources, as defined in the paragraphs that follow). Composite or aggregated resources can be located in more than one machine.
- Resources are also described with respect to their location, depending on the location of the machines they reside on.
- Resources have specific QoS properties, each property being measured by its corresponding unit of measurement.

The distinction between aggregated and composite resources aims to distinguish between heterogeneous and homogeneous sets (bundles) of resources:

- A Composite Resource is a resource that comprises at least two heterogeneous resources (e.g. a Compute Node comprises a CPU and a Hard Disk).
- An Aggregated Resource is a resource that comprises at least two homogeneous resources (e.g. a Cluster comprises more than one Compute Nodes).

As already mentioned, currently we distinguish between two main types of resources: computational and storage resources.

- A Computational Resource is a Composite Resource that comprises at least one CPU, a volatile memory and an operating system. A Computational Resource can reside on at least one machine (we can have two CPUs in different machines).
- A Storage Resource can be a Hard Disk or any other type of permanent storage. A Storage Resource resides on a single machine. In case of representing a bundle of Storage Resources, we introduce the class Storage Aggregated Resource. Different Storage Resources that comprise a Storage Aggregated Resource may reside on different machines.

However, not all the resources are tradable: For instance, one may not trade a CPU or a Hard Disk, as such. Tradable resources include Compute Nodes and Clusters.

- A Compute Node is a type of Composite Resource that comprises exactly one Computational Resource and any number of Storage Resources.
- A Cluster is an Aggregated Resource comprising a set of Compute Nodes.

Orders and Offers

As already mentioned, consumers and providers specify their needs for the trading of resources by placing market orders: These are either consumers requests or providers offers.

A Request describes the resource needs of a consumer (quantity of requested resources and time intervals of their allocation) and the price she is willing to pay for those resources within a specific Market. An Offer specifies the exact resources (quantity of offered resources) that a provider trades in specific time intervals and price: This is in contrast to the specification of requested resources, where the consumers may request a class of resources. Offers specify the exact individuals that providers sell.

In a Grid market, multiple, alternative orders of the same type (offers or requests) may be connected using an XOR (exclusive OR) connective.

Bundle orders allow consumers and providers to indicate their needs for composite resources. Although in several systems it could be adequate to use bundles of heterogeneous resources (i.e. Composite Resources) from different providers, this approach could not be useful in terms of QoS. Thus in the proposed approach, a single bundle unit is represented by a Compute Node entity as an atomic unit of resource allocation through markets. A more complex bundle can specify a set of Compute Nodes satisfying certain qualities, which are further explained below.

When consumers and providers have to allocate resources within a period of time, their correspondent orders (requests/offers) need to allow the specification of time intervals delimited by the order-end-time and the order-start-time specifications. Doing so, consumers (providers, respectively) may specify a time range within which resources have to be allocated. In addition, they may need to allocate resources for some time slots (whose size is being explicitly specified) within this time interval. For the consumers, in most of the situations, may be indifferent when the allocation takes place: They may only require acquiring the resources before the given deadline.

Furthermore, the specification of offers and requests must provide flexibility to expressing conditions of trade. For example, a consumer may need to specify whether the allocated resources come from the same provider or from different providers. Generally, in case that a bundle of resources has to be allocated, the minimum or maximum number of providers may be specified, together with the agent(s) by whom a particular order has been issued (property ordered by). Also, the initial price of a traded resource (for offers this is the minimum price and for requests this is the maximum price) is specified by the property price-of-resource.

A snapshot of the taxonomy of the developed ontology can be seen at figure 3.2. The ontology has been developed following a collaborative approach to the engineering of ontologies [53].

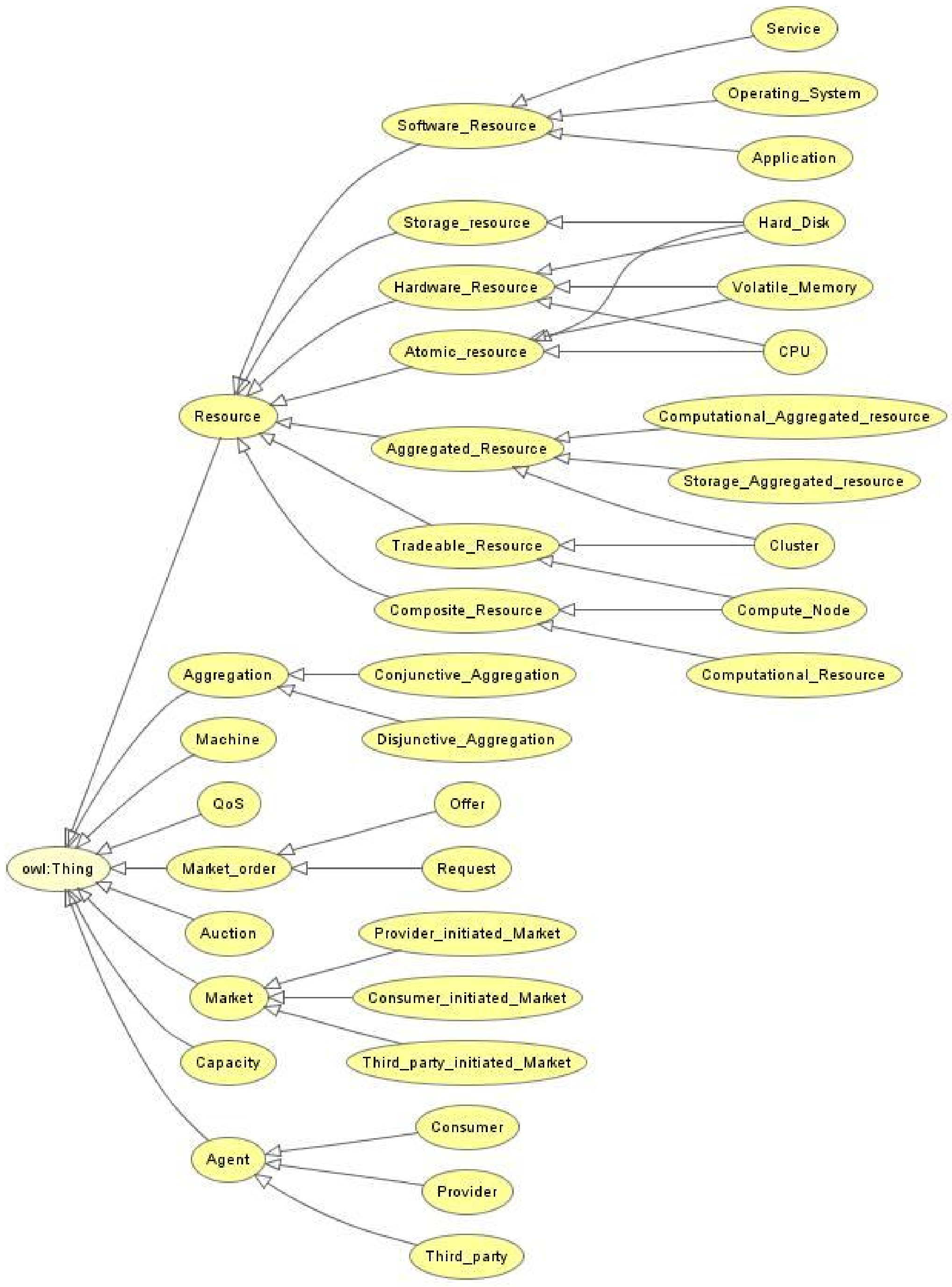


Figure 3.2: The resources ontology

3.5 From semantics to a language

The semantic approach presented in the previous sections will be used along the rest of the chapter as a semantic basis for the bid and offer specification. The specification is motivated by the need to enable flexible bid and offer configuration in dynamic environments such as Contributory Systems. Concepts defined in the ontology will directly be mapped to concepts in the bid and offer specification language introduced next. As already stated, Grids are environments characterized by the heterogeneity and diversity of their resources, applications, applications behaviours, dynamics and scale where not a single auction is able to efficiently deal with such issues. Thus, recent research [5, 15, 66] propose the cohabitation of multiple market mechanisms to mediate resource allocation in Grids.

In this chapter we deal with flexible bidding and offer configuration for open resource marketplaces where multiple market mechanisms cohabit. In those environments buyer agents need to be able to describe their preferences for resources that possibly are being sold by different auction types. Furthermore, seller agents need to set their initial offers to be traded by specific market mechanisms. On the one hand a desirable property for bidding specification is that of a common bid/offer specification language that facilitates bid/offer description independently of the market mechanism used to allocate the item. On the other hand, the bidding specification may facilitate the winner determination problem resolution by providing an efficient organization of information.

Combinatorial auctions are efficient mechanism devised for the allocation of bundled items that only require an efficient data structure to represent user's preferences. In contrast, single item auctions such as English or Double auctions; require specific item descriptions since they are only able to trade in multiple units of one specific item. Given a common bidding specification either for combinatorial auctions and single item auctions, we recognize the need for a set of bidding support tools that facilitate the bidder task when dealing with multiple market institutions.

3.6 A Formal Bidding Language

The objective of our work is to develop a complete bidding specification that is supported by any type of auction, that is, allow bidders to formulate bids able to be submitted to any auction instance. The bidding specification should provide semantics to let bidders formulate their preferences for Grid resources. Buyers and sellers preferences can be complex requiring combinations of items subject to time and quantity preferences and other requirements for full or partial satisfaction. All this requirements can also be constrained by different clauses such as timing or workflow relations amongst sub-bids.

In the following section the descriptive language is presented, the language allows to express bidder preferences for Grid resources independently of the type of auction. Later in section 3.10.1 some tools to automatically adapt a bid to different types of market are introduced.

3.7 Related Work on Bidding Languages

Bidding languages have focused on the compact specifications of bidder's preferences in mainly combinatorial auctions. This is also due to the fact that the optimization problem (to generate the matching and winning bids) is NP-Complete and hence heuristics are required to reduce the length of the problem by providing means to express compactly the preferences. "Logical Bidding Languages" rely on standard logical operators to represent user's preferences [67, 78]. OR bidding languages allow bidders to define non-overlapping bids for a set of resources but not for substitute preferences. In these languages, bidders may face a budget exposure problem[1]. XOR languages, allow bidders to express substitute preferences but all bids from a bidder are mutually exclusive. When a bidder wants any combination of items, he must explicitly bid on each combination, so the process is not scalable since it requires the expression of an exponential number of combinations

[1]From an economic perspective, the exposure problem can lead to inefficient assignments in two different ways. First, a bidder who decides to bid aggressively may acquire unwanted items. Second, a bidder who decides not to take the risk may fail to acquire items for which it actually has the highest valued use. In each case, the assignment is inefficient because items are not put to their best use.

for even a small number of items. OR* languages provide more expressiveness by means of "phantom variables" however such languages may not be clear to bidders on how to express their requirements [39].

Hoos et al. [45] presents the $\mathcal{L}_{GB}$ language that allows the combination of goods with AND, OR and XOR operators. They also extended the language adding the clause *k-of* that allows bidders to express willingness for some k items within a subset. $\mathcal{L}_{GB}$ provides complete expressiveness to bidders but requires the construction of entire trees to bidders that may not know how to express their requirements. As far as we know the most recent work on bidding languages is TBBL, a tree based bidding language for combinatorial exchanges [19]. TBBL allows double sided exchanges, that is, not only provides semantics for the allocation of goods but also for the re-allocation. TBBL has been designed to be concise and structured. It allows for specification of both bids and asks in a single structure, and allows agents to specify upper and lower bounds on their values for trades. TBBL also provides a compact way of representing bids in the tree. As our understanding, TBBL trees will not be portable within different market structures. That is, given a formulated bid, it requires a pre-process to adapt bidder's preferences to specific market requirements.

The ClassAd language [74] can be considered a precursor of our work, their authors set the need for generic and configurable bidding specification so as to enable the description of heterogeneous resources in the Grid. ClassAd provides semantics to flexibly specify resource requirements for both sellers and buyers, enabling also the specification of constraints. However, ClassAD is not a structured language based on logical operators. This fact, does not constitute a limitation but makes the language less appropriate to express complex bids.

3.8 A Tree Based Bidding Specification

Our envisioned bidding specification language represents bids in form of a tree where internal nodes contain logical operators (OR, AND or XOR) and leaf-nodes contain specification of bidder's requirements. The root node is always a XOR constraint and internal nodes may contain any of those logical operators.

The use of logical operators is relevant for the bidding language because enables users' to express complex requirements and the elicitation of different preference. XOR constraint allows bidders to express substitute preferences over resources. For example a bidder may want access to either one CPU of 2GHZ during an hour or access to two CPUs of 2GHz for half an hour each one. By means of OR operators bidders indicate their willingness to accept partial satisfaction whilst AND operators indicate their requirement for complete satisfaction. One key aspect of our bidding specification is that bids can be expressed in a compact manner or in a complete manner. Figure 3.3 shows the same bids expressed in both manners.

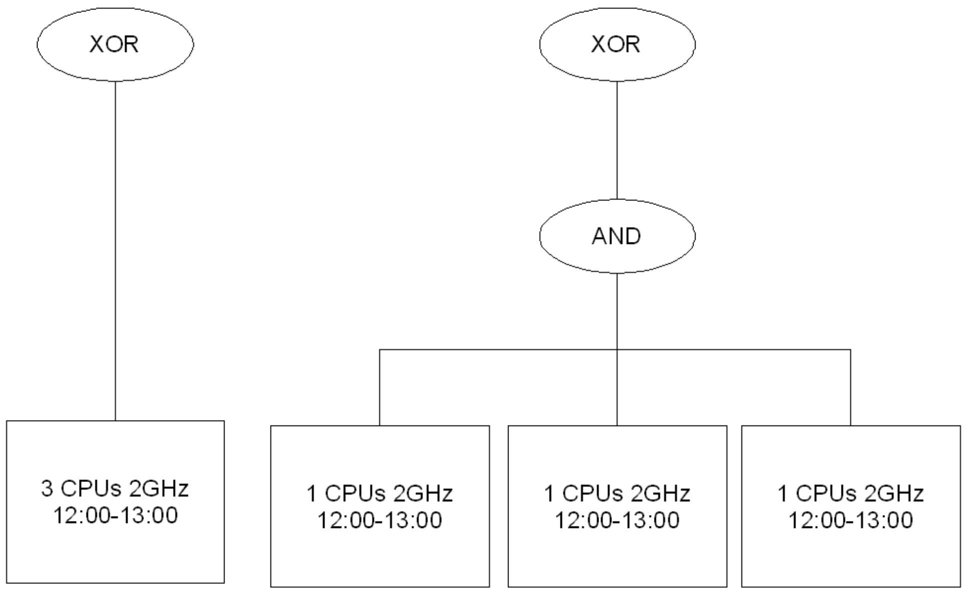


Figure 3.3: Compact vs detailed representation of a bid.

Code 1 Definition of a resource.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
 <xs:include schemaLocation="./TradedItem.xsd"/>

      <xs:complexType name="SimpleResourceType">
      <xs:sequence>
        <xs:element ref="TradedItem"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AggregateResourceType">
      <xs:sequence>
        <xs:element name="SimpleResource"
                    type="SimpleResourceType"/>
         <xs:element name="quantity"
                    type="xs:nonNegativeInteger"/>
      </xs:sequence>
    </xs:complexType>

  <xs:complexType name="CompositeResourceType">
        <xs:sequence maxOccurs="unbounded">
        <xs:element name="AggregateResource"
                    type="AggregateResourceType"/>
        </xs:sequence>
  </xs:complexType>

  <xs:element name="Resources">
   <xs:complexType>
      <xs:choice>
         <xs:element name="CompositeResource"
                    type="CompositeResourceType"/>
        <xs:element name="AggregateResource"
                    type="AggregateResourceType"/>
       <xs:element name="SimpleResource"
                    type="SimpleResourceType"/>
      </xs:choice>
    </xs:complexType>
    </xs:element>
  </xs:schema>
```

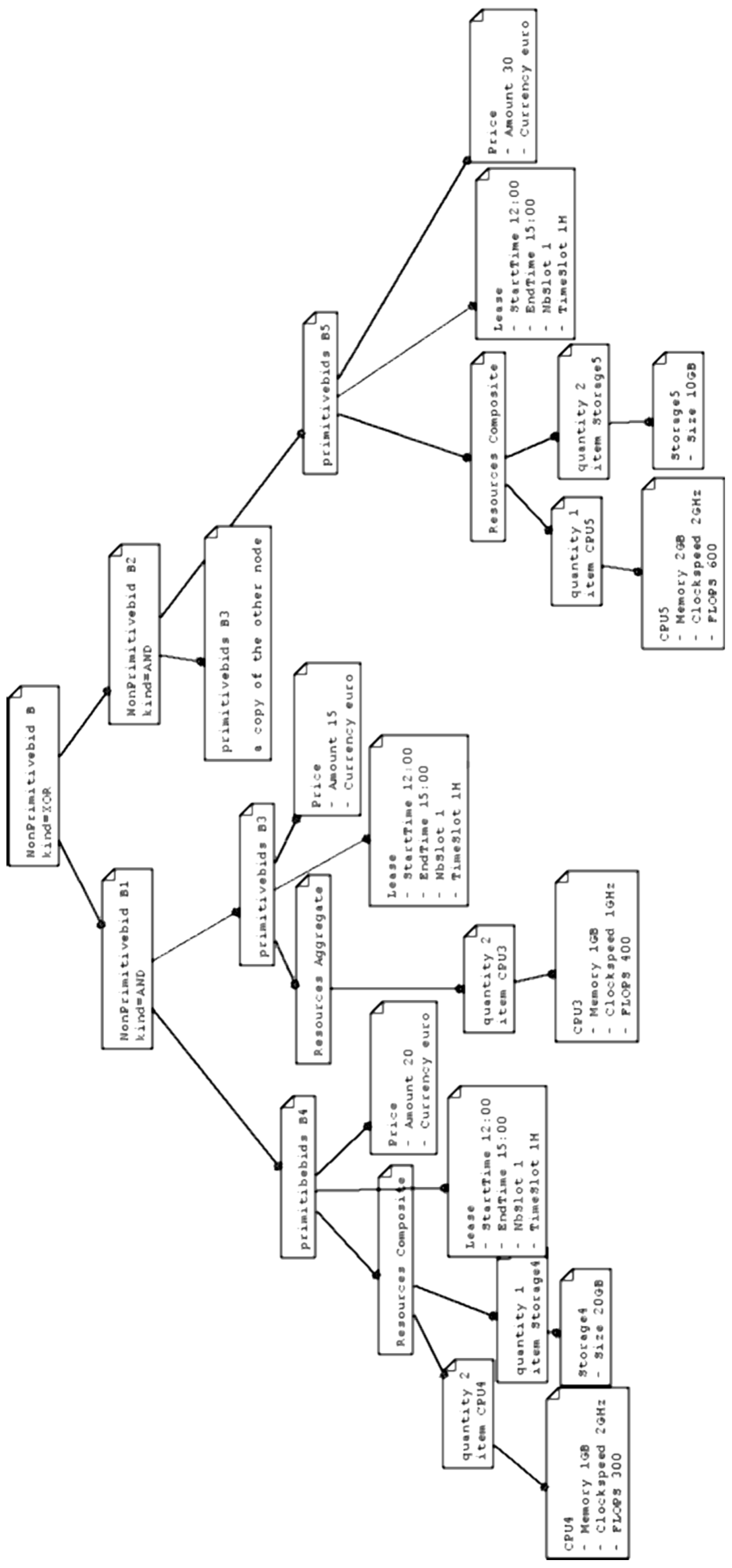


Figure 3.4: Tree representation of the example bid.

3.8.1 Leaf-Node Specification

Leaf-nodes are used to capture the resource specification. One requirement of the bidding specification is that of support to both seller's preferences and user's preferences, that is, the same specification would be used to formulate bids and asks. A leaf node is composed of four different items.

Resources: Three notions for resource specification have been introduced following the semantic approach provided by the ontology, i.e. composite resources, aggregate resources and simple resources. A simple resource represents one unit of a type of resource, termed item. An aggregate resource represents multiple units of simple resources whilst a composite resource represents a bundle of different resource types. These concepts provide flexibility to the market initiators so that they can easily decide which the product that they will offer in the market is. Furthermore, aggregated resources may be conjunctive, that is, the traded item is an atomic aggregation of multiple units of a basic resource that may be allocated entirely. In contrast disjunctive aggregated resources only require a subset to be allocated. This differentiation addresses the AND and OR notions introduced before as well. For example, a compute node is a composite resource (CPU + Storage) and four compute nodes are described as a conjunctive or disjunctive aggregated resource depending on the type of satisfaction required. A code sniped of the resource specification can be seen in Code 1 Figure. A simple resource may be either storage or CPU each one characterized by its properties which can be seen in Code 2 Figure.

Lease: A lease is defined by the start time, the end time, the number of required slots within the time interval and the size of the time slot. Note that leases may be precise in the sense that the bidder requires the same number of slots as the interval has. However, the lease may be imprecise, requiring a minor number of slots than the time interval has.

Price: Price is defined by the amount of some currency that the user is willing to pay for the resource description and the specified lease time.

Constraints: In addition, bidders may add some constraints to the bid like some imprecise quantity requirements. In our bidding specification ”at least” and ”at most” clauses are allowed so as to enable bidders to imprecisely specify upper or lower bounds in their requirements. Next section discusses other workflow related constraints.

Code 2 Definition of an item.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
  <xs:element name="TradedItem">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="ResourceID"/>
        <xs:element ref="ResourceType"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="ResourceID" type="xs:nonNegativeInteger"/>
  <xs:element name="ResourceType">
    <xs:complexType>
      <xs:choice>
        <xs:group ref="Storage"/>
        <xs:group ref="CPU"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:group name="Storage">
    <xs:sequence>
      <xs:element ref="Size"/>
      <xs:element ref="Throughput"/>
    </xs:sequence>
  </xs:group>
  ...
  <xs:group name="CPU">
    <xs:sequence>
      <xs:element ref="Memory"/>
      <xs:element ref="ClockSpeed"/>
      <xs:element ref="FLOPS"/>
    </xs:sequence>
  </xs:group>
  <xs:element name="Memory">
    <xs:complexType>
       ...
    </xs:complexType>
  </xs:element>
       ...
 </xs:schema>
```

3.8.2 Implementation

The bidding language is specified as an XML encoded schema that represents the resources required(offered) by buyers(sellers) and the preferences. A bid can be looked upon as a tree where the non-leaf nodes represent operators such as AND, XOR, OR, and the leaf nodes specify the exact item (resource) description. An item specified at the leaf node should correspond to a resource item traded at the market or as mentioned before be an imprecise item description.

In the XML schema we refer to the non-leaf nodes as NonPrimitiveBids and the leaf nodes as PrimitiveBids. The leaf node includes attributes to indicate the desired quantity of the item, the leasing attributes (start time, end time, duration of time slot, and number of required time slots (for imprecise bids)) and the price. The item may be either a simple resource, a composite resource, or an aggregated resource. Each resource is of one type (CPU, Storage,...) that encapsulates the main attributes of that type of resource. Non-primitive nodes represent the relation between its children nodes, i.e., XOR, AND, OR. Figure 2 and Code 3 present a example bid for one of two configurations of resources to be used for a specified period of time (3 hours from 12:00 to 15:00). The example bid expresses that the buyer requires one of the following: one CPU of 400 FLOPS, 2 CPUs of 300 FLOPS, or one CPU of 600 FLOPS.

Code 3 Textual representation of an example bid

```
B=(B1 XOR B2)

B1=(B3 AND B4)

B2=(B3 AND B5)

B3: 1CPU(1GB, 1GHZ, 400FLOPS) FROM 12:00
   TO 15:00 FOR 1 TIMESLOT WHERE A
   TIMESLOT = 1HOUR. I will pay 15.

B4: 2CPU(1GB,2GHZ,300FLOPS); STORAGE(20GB)
    FROM 12:00 TO 15:00 FOR 1 TIMESLOT
    WHERE A TIMESLOT = 1HOUR. I will pay 20.

B5: 1CPU(2GB, 2GHZ, 600FLOPS); 2 STORAGE(10GB)
    FROM 12:00 TO 15:00 FOR 1 TIMESLOT
    WHERE A TIMESLOT = 1HOUR. I will pay 30.
```

3.9 Workflow specification

In some situations a bidder may want to express relations within its bid requirements. In our schema we defined a set of relations amongst bids to allow such type of workflow specification. Six clauses have been defined:

R <enables> X this clause is used to specify the willingness to get X only if R can be obtained. However does not ensure that getting R, X will be obtained. In any case, the allocation of R is independent of getting X.

R <not after> X Used to specify precedence in time of some requirements. Time ranges from R and X should overlap but this constraint will ensure that R will be allocated before X. However if X is allocated and not R, the clause does no affect.

R <not before> X Contrarily to the previous case, used to express precedence of X to R.

R <only if> X In this case, R will be allocated only if X is allocated. This clause only has effect when R is allocated requiring to wait to know the allocation of X.

<at Least> Q of T in R, MAX A quantity Q of an attribute T of a resource R is at least required. This sets a lower bound for that attribute. The constraint also requires to specify a maximum quantity.

<at Most> Q from T in R A quantity Q of an attribute T of a resource R is at most required, This sets an upper bound for that attribute. At most indicates partial satisfaction and can also be expressed as an OR collection of requirements.

3.10 Support for multiple auction formats

Generally bidding languages have been developed to support one type of auction, that is, they organize bids in data structures that facilitate the WDP (Winner Determination Problem) resolution for a specific auction setting. One of our intentions was to provide multi-auction bidding support able to hold up multiple types of auctions (from single sided auctions to combinatorial auctions). However, this does not of course imply that every market should support the complete bidding specification but only a subset of the language. For example, a market employing an auction mechanism that cannot guarantee complete allocation of a request will not accept AND operators in the bid tree.

	Combinatorial	Double Auction	Iterative
Item	Bundle and single	Single	Single
Units	Multiple	Single*	Single
Time	Multiple	Single*	Single*

Table 3.1: Requirements for the WDP.

Table 1 presents the characteristics of the items able to be traded by different auction mechanisms. Some auction models, in particular combinatorial auction models resolved using mixed or integer programming techniques allow specifications of constraints. Thus bidders can issue requests such as requiring bundles of items for 2 continuous hours of computational capacity between 12:00 and 19:00. Such requests may nevertheless be imprecise in other auction formats like iterative auctions and DA that neither allow the trading of multiple time slots as individual units nor imprecision in bid time ranges. They contrarily, require selling items as a whole and for precise periods. Thus, multiple time slots need to be allocated by multiple auctions.

Therefore, the bidding specification presented in section 5 can be directly used to represent bids for combinatorial auctions because the WDP is able to handle imprecise bids. In contrast, DA and iterative auctions require multi-item bids to be pre-processed into single item bids as defined by the auction setting.

3.10.1 Bid decomposition

In the later section the need for pre-processing multi-item bids into single-item bids has been identified. Pre-processing may be looked upon as an internal process that returns semantically equivalent bids but able to be handled by a specific allocation mechanism. There is the need to point out that a given market (instance) regulates the operators that may be present at non-leaf nodes. For example, a market employing an auction that

[1] *The clearing process requires items to be single

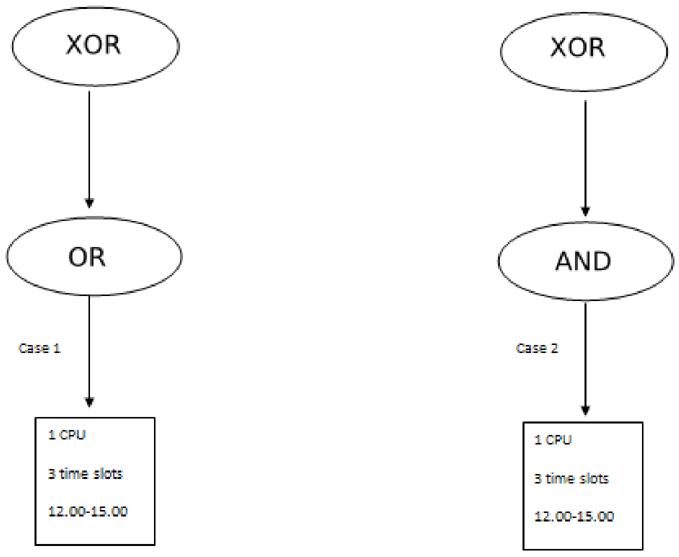
cannot guarantee the complete allocation of the request will not accept the AND operator.

The following example will be used to illustrate the different possibilities of decomposition:

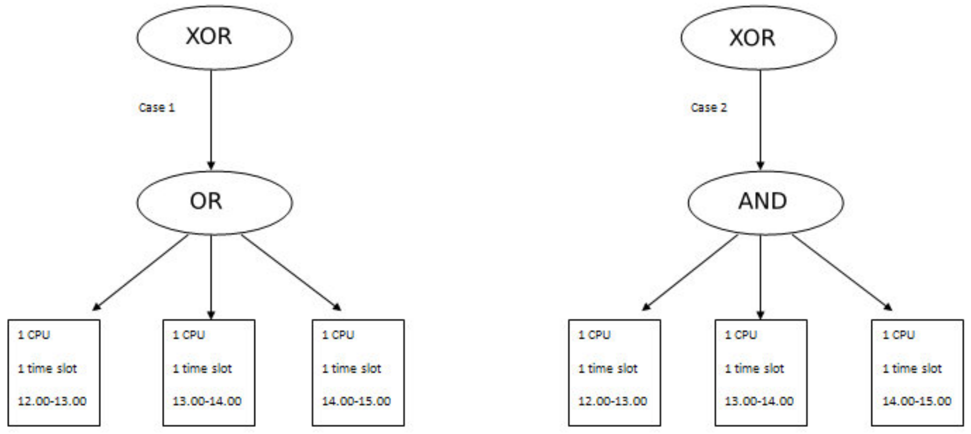
Let's consider an auction that is trading one CPU of 400 FLOPS for the time range compressed within 9:00 and 19:00 where each time slot is 1 hour of duration.

The bids that users are able to formulate are of the following types:

1. **Exact preference in quantity and time:** A bid B3 requires one CPU of 400 FLOPS for 3 hours from 12:00 to 15:00, that is, the bidder is asking for a precise time range. Case 1 of figure 3.5a shows the compact bid representation, that is the way user formulates the bid. Note that for this example bid partial satisfaction is required (OR constraint). Case 2, presents the same case when complete satisfaction is required (AND constraint). Figure 3.5b represents the same bid but showing it fully pre-processed. Figures will present either compact and full decomposition trees except for the cases where the size of the full decomposition prevents from a clear understanding.



(a) Compact representation



(b) Detailed representation

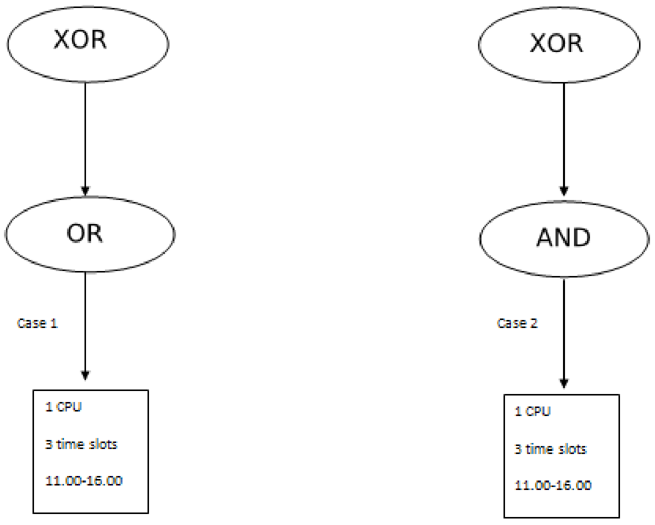
Figure 3.5: Exact preferences in quantity and time.

2. **Exact preference in quantity but not in time:**

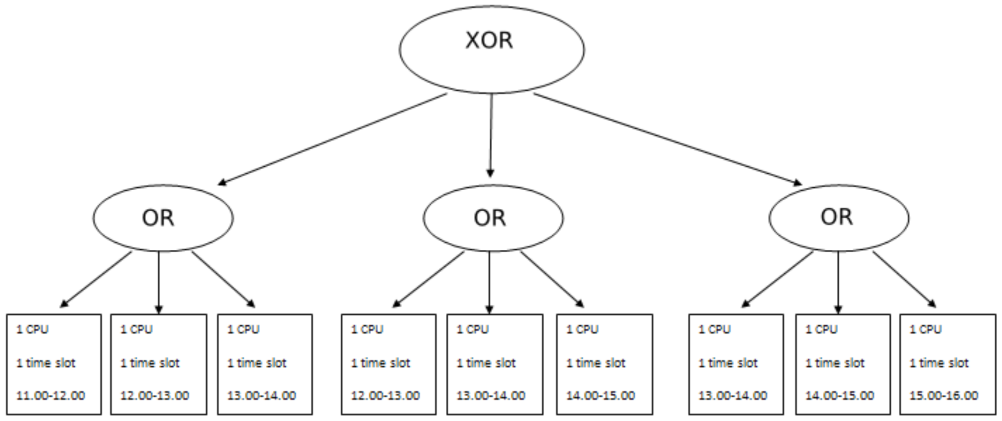
   (a) **Require time to be consecutive:** A bid B4 requires one CPU of 400 FLOPS for 3 consecutive hours within 11:00 and 16:00. This case requires the formulation of one XOR bid with (*number of available slots* −

number of required slots + 1) AND (OR for the case of partial satisfaction bids) sibling bids each one with 3 precise leaf nodes. Case 1 of Figure 3.6a shows the bid formulated by the bidder for the case when partial satisfaction is required. Case 2, presents the same case when full satisfaction is required. Figure 3.6b presents the tree completely pre-processed for the case of partial satisfaction.

(b) **Do not require time to be consecutive:** A bid B5 requires one CPU of 400 FLOPS for 3 *any* hours within 11:00 and 16:00. This case requires to pre-process the bid into a tree with $\kappa = C_n^m$ leaf nodes. Due to its size a figure is not provided, however the case is very similar to the one presented in figure 3.6. Indeed, the compact representation is the same as the figure 3.6a.



(a) Compact representation



(b) Detailed representation (only for the partial satisfaction case)

Figure 3.6: Exact preferences in quantity but not in time constrained to be consecutive

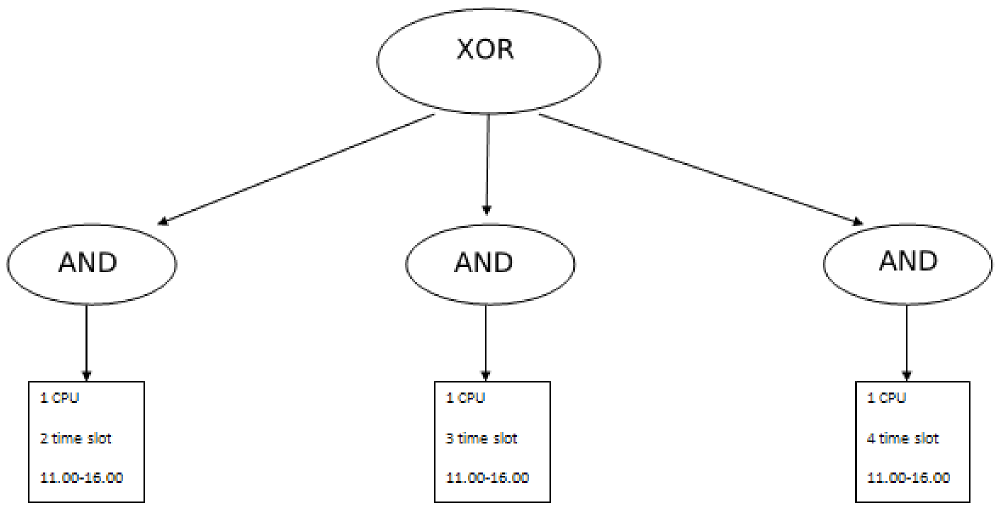


Figure 3.7: Exact preferences in quantity but not in time. Time is not constrained to be consecutive.

3. **Neither exact preference in quantity nor in time:**

   (a) **Require time to be consecutive:**

      i. **By excess:** A bid B6 requires one CPU of 400 FLOPS *at least* for 2 consecutive hours within 11:00 and 16:00. In this case, the auction provides complete satisfaction (the auction only accept AND operators), otherwise does not make sense the mandatory 'at least'. The decomposition needs to generate a bid tree of XOR(root node) and AND due to impreciseness in time (multiple possibilities). The maximum bound on number of time-slots must be set by bidder. Due to the size of the decomposition, the figure 3.7 only show the decomposition of non-leaf nodes whereas leaf nodes are kept in a compact representation. For the example, the bids are decomposed into precise bids for 2 time slots, 3 time slots and 4 time slots (bound set by the bidder). The complete decomposition (including leaf nodes) would be similar to the decomposition showed in figure 3.6b.

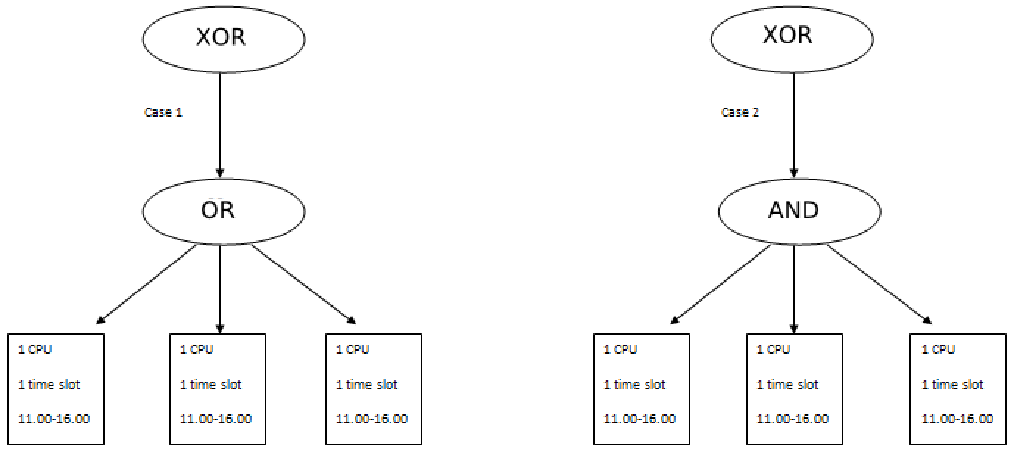


Figure 3.8: Neither exact preference in quantity nor in time. Each leaf node is split in: ($number\ of\ available\ slots\ -\ number\ of\ required\ slots\ +\ 1$) bids.

ii. **By default:** A bid B7 requires one CPU of 400 FLOPS *at most* for 2 consecutive hours within 11:00 and 16:00. This case is simpler and similar to the case illustrated in Case 1 of figure 3.5. Notice that the requirement 'at most' indicates partial satisfaction which is expressed with an OR constraint.

(b) **Do not require time to be consecutive:**

i. **By excess:** A bid B7 requires one CPU of 400 FLOPS for at least 2 any hours within 11:00 and 16:00. The case is similar to the one illustrated at figure 3.8. Though, the number of leaf nodes increases, in fact, the possible combinations are $\kappa = C_n^m$ where m represents the number of requested time slots and n represents the total number of available time slots. The decomposition problem here requires to generate an exponential number of bids, however the bidding language allows bidders to represent their specific willingness for dis-contiguous time slots by means of AND operators and not only expressing an "at least" condition.

ii. **By default:** A bid B8 requires one CPU of 400 FLOPS for at most 2 any hours within 11:00 and 16:00. This case needs to generate again $\kappa = C_n^m$

leaf nodes. The tree will be represented as XOR node in the root, $\frac{\kappa}{n}$ OR nodes as siblings where each sibling has m leaf nodes. The case however, should be simplified by the bidder requiring specific time slots constrained by the OR operator.

3.10.2 Integration

The bid decomposition utilities have been developed as a *BidManagement* component developed using the Fractal component model with Java language mapping that offers a set of functionalities to decompose bids. The component approach facilitates the integration with any auction platform. Composition can be done by static specification through the ADL (Architecture Description Language) that describes the system composition and binding of sub-components. Alternatively, standard interfaces like Web Services can be used by any other component to access the offered set of functionalities.

The decomposition functionality requires two parameters, the bid formulated by the bidder using our bidding language and the description what an item is for an specific auction setting. The output for the process is a bid tree semantically identical as the one formulated by the bidder but following the structure required by the auction setting.

3.11 Evaluation

We aimed to evaluate the time required for the bidding support tools to decompose bids. The cost of decomposition will be directly related to the size of bids and the required combinations of time slots. Our experiments were conducted in a 3Ghz pentium IV processor with 512Mb of memory.

The aim of the evaluation was to determine whether the proposed bidding language and the decomposition tools could be used in Grid marketplaces without introducing to much overhead. It is clear that the amount of time taken to decompose complex bids must not exceed the time that an auction takes to clear. While for scheduled auctions this does not seem an important problem since they have their well defined phases (i.e. the bidding phase is scheduled before the clearing phase) and consequently the time taken to decompose bids can be considered by the administrator of the system. Continuous auctions must not be hindered by the effects of bid decomposition. In that sense, we consider that the entire process must not take times over the order of seconds.

In the experiment we generated a test XOR bid for one item requiring different number of time slots. Namely we generated 24 different bids each one requiring a different amount of time slots, from one to twenty four. We measured the time taken to decompose the bid into an equivalent bid suitable for an auction selling different items. The auctions chosen offered items for 24h but at different time slot granularity. That is, one auction offers any time slot from 0h to 23h where the time slot has 1 hour of duration while another offers any time slot from 0h to 23h where the time slot is of 3 hours of duration. Thus, we used different time slot granularities as can be seen in figure 3.9.

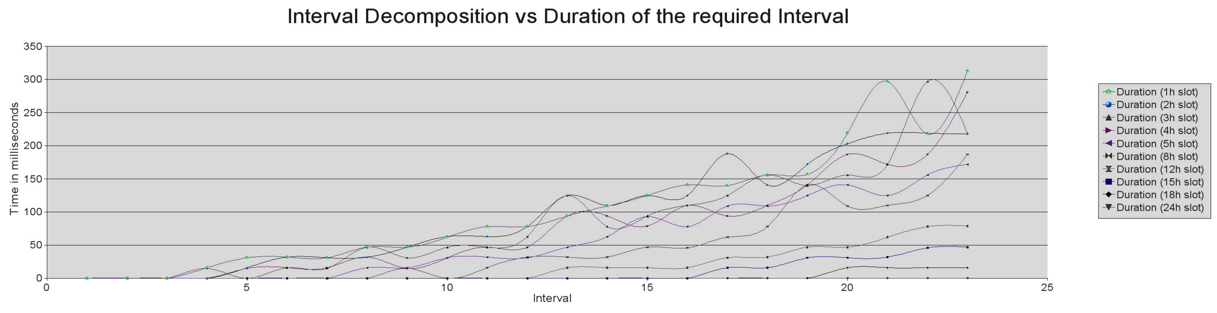


Figure 3.9: Time taken to decompose imprecise bids into precise bids. Each line represents a different bid requiring a different time slot granularity.

In figure 3.9, we can look at the line that represents those bids suitable for auctions that offer items in slots of 1h. X-axis indicate the number of slots required by the bid, while Y-axis shows the time in milliseconds taken for the computation. As it can be seen, bids for either one, two or three time slots are decomposed with a time near zero. A bid requiring 9 time slots which is expressed for example as 12:00 to 21:00 is decomposed in 9 bids of one 1h time slot each one in nearly 50ms. As expected we can see that as the number of slots increases the time taken to decompose bids also increases.

Other lines show the cases when bids have to be decomposed in other than one hour-sized time slots. For example the line that represents those bids that will be submitted into markets that offer items of eight hours time slot granularity. In this case, decomposing a bids for 16 time slots into two bids for 8 time slots each one will take approximately 50ms.

As the size of the time slot increases the time for computation is reduced since the number of possible combinations is also reduced.

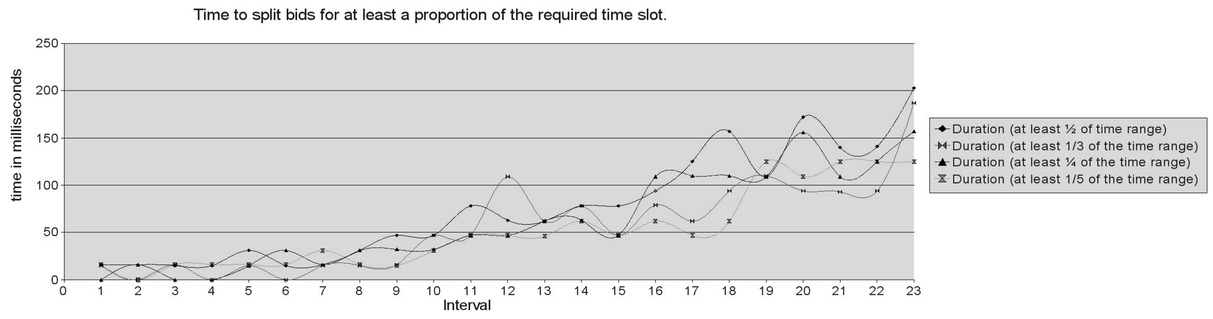


Figure 3.10: Time required to decompose bids requiring an interval of time. Each line represents a different bid requiring a different time slot granularity.

Figure 3.10 shows the time taken to decompose imprecise bids into precise bids. Four different types of imprecise bids were generated. Each one is defined for a time range requiring at least a half, a third, a fourth or a fifth part of the required time range respectively. That is, a bid within a time range compressed between 10:00 and 20:00 that requires at least a half of the time slots will require at least 5 time slots. X-axis indicates the interval for which the bid was submitted, the bid from the example above will have an interval of 10 time slots.

As it can be seen in the figure 3.10, the wider the time range the greater the time required to decompose bids. Besides, the higher the portion of the time range required the higher the cost of computation. This seems to be incoherent with what decomposition is doing since as smaller proportion within a time range more possible combinations exists. However each combination is of small size. This behaviour is attributed as the way decomposition have been programmed. Particularly, the cost of calculation of all possible combinations is not the relevant part but the creation of precise nodes. Possible combinations are built by constructing a tree of pointers to the generated nodes avoiding to create multiple copies of them. Thus, as more required nodes, higher is the cost to process them.

Other types of decomposition such as those that decompose bids in units can be compared to the results of figure 3.9.

3.12 Conclusions

The chapter characterized Grid resources with the aim to provide a bidding language able to be supported by multiple auctions without any modification. The contributions of our presented work are twofold: Firstly, an expressive Grid oriented bidding specification language developed as an XML schema. Secondly a set of functionalities that provide automatic bid pre-processing for the adaptation of bidders imprecise preferences to different types of auction settings. The chapter presented an evaluation of these tools which introduce a cost that does not take more than hundreds of milliseconds in the worst case. The bidding language and support tools have been successfully used at the Grid4All project for the resource allocation framework. We have learned several things from this work. First, Grid resources are heterogeneous in nature which requires any bidding language to be flexible and extensible. Second, bidding specification can be complemented with a set of support tools to enable generalization of the language while keeping its precision and flexibility. Support tools can be offered, like many others, as a replicated service of the Grid. Thus, bidders may not be required to have a knowledge of the mechanism they are using to obtain resources. The presented work improved current existing languages by enabling transparent support for different auction formats while keeping a structured representation of the information.

Chapter 4

Architectural Approaches for Resource Allocation

4.1 Introduction

VOs are autonomous entities that are created to support services used by a community; a key objective addressed is how to expand the computational capacity and attach computing resources when needed. Utility providers practice posted-price mechanisms, but this does not address fluctuations in supply and demand. QoS based mechanisms are complex to implement and require the consideration of multiple variables to become efficient increasing the complexity of the system. In contrast, dynamic pricing market mechanisms are able to adjust supply to demand through a single variable, i.e. price. In a possible scenario, consumers, in that case virtual organisations, have requirements to execute jobs or make use of certain quantity of resources. Providers, either commercial entities, Virtual Organizations or individuals on the Internet specify times when their resources are available. Market-place operators will use auction-based mechanisms to allocate jobs to resources.

Economic models are widely used to allocate resources, but which model or mechanism is the most appropriate for a particular situation is not clear. In open environments applications may have diverse demand functions and resources are heterogeneous. Elastic applications such as video transcoders adjust to variable resource quantities. These may be satisfied through mechanisms such as k-double auctions. Data mining applications may need both computational and storage resources; acquiring one without the other is not acceptable. They have complementarity that may be handled only by combinatorial auctions. Some applications may also be able to make trade-off in different dimensions, for example sacrifice memory for CPU or vice-versa. This requires auction formats that interact with trading agents and progressively elicit preferences and in particular for substitute resources.

Closed and controlled Grid environments may be best supported by centralized and persistent markets that clear and compute the optimal allocations either continuously or as scheduled events. Open environments such as Contributory Systems need to handle spontaneous creation of markets on-demand, where the creators (could be buyers, sellers, mediating brokers) choose market models and structures that best suit their needs. Virtual organizations running real-time applications that (de)allocate resources on-demand, through for example, continuous monitoring of load, may require a market structure that clears continuously. Applications that plan their resource needs may best be satisfied by markets that schedule their clearing. Network sensitive virtual organisations may have preferences to maintain locality in resource allocation. A natural way is to start a market that restricts the participating sellers to those whose resources satisfy proximity constraints.

This chapter addresses the main aspects to be considered for the design of an architecture to mediate the resource allocation in Contributory Systems. A set of properties have been identified that have to be considered when designing such an architecture. Second, the chapter proposes a peer-to-peer architecture to handle allocation of resources amongst Virtual Organizations attaining the identified properties. Specific resource allocation will be managed by auctions. Auction mechanisms will be handled by generic components that offer the main functionalities to support different auction institutions. Support for multiple market mechanisms is required so as to support diversity of supply and demand which is one of the main characteristics of Contributory Systems. In those systems, Virtual Organizations with different aims may want to expand their capacities through bidding on markets, as they are different organizations with different purposes, their needs are not the same so they won't require neither the same type of resources nor the same type of allocation. As a result, a single market mechanism may not be useful but several different market institutions may be required. This fact motivates the built of a generic auction container able to be configured and deployed according to initiator needs.

4.2 Requirements

Based on the motivation presented in the introduction of this Chapter, a set of requirements for a configurable resource allocation framework are derived. Some requirements are addressed by architectural choices while others by technological choices. This requirements have been identified taking into account the main characteristics of the addressed scenario. As a remainder, Contributory Systems are heterogeneous and dynamic systems where resources are provided by their participants in favour of the community. Heterogeneity and dynamism require flexibility and configurability of the system so as to support the wide range of possibilities. Derived from that, the following set of requirements have been identified:

- **Generic Infrastructure**: Applications in Grid are heterogeneous, a single market mechanism will not provide efficient allocations for every possible scenario. A generic framework will set the basis to support multiple market mechanisms. It can be obtained by means of a structured design, separation of concerns and generic APIs.

- **Scalability**: The framework should not limit the scalability of the system. Decentralization and distribution should be considered basis for the design.

- **Dynamicity**: Due to the nature of Grid, ever running and ever changing, the framework should allow dynamic deployment and configuration.

- **Open architecture**: Grids are continuous evolving systems. An open architecture that can be extended and upgraded is mandatory for a long-living market place.
- **Standardization and Interoperability**: The use of flexible standards and interoperable interfaces to facilitate interaction with external infrastructure services is also required.
- **Decentralization and self-organization**: In case of connections, disconnections and failures, the system should keep functioning (it shouldn't have a single point of failure) and should reorganize without requiring any external intervention, getting to a consistent state as soon as the available resources and VO stability allow it.
- **Individual autonomy**: The VO's members should be free to decide which actions to carry out, what resources and services to provide, and when to connect or disconnect.
- **Market availability**: Market services should always be available (if needed) as long as there are enough resources to execute them in the VO.
- **Location transparency**: Market services don't have to worry about other market service's location. The system resolves them transparently, and services access each other using a location-independent identifier.

4.3 Dynamic Market Deployment for Decentralized Resource Allocation

In this section we present DyMRA[1], 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

[1]Dynamic Market Deployment for Decentralized Resource Allocation

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.

4.3.1 Scenario

The scenario presents three different VOs A, B and C that provide general purpose functionalities such as a file sharing services and communication services to their members. VO A is an online gaming community where few members contribute regularly their computational resources to the community; instead members pay a subscription fee to obtain the services. B is a scientific community and its purpose is the sharing of knowledge and technical documents amongst its members; generally its members contribute with their resources to the community. Finally, VO C is a photo sharing community where members usually contribute with their resources.

This work focuses on the allocation of resources provided by Virtual Organizations to other Virtual Organizations and fits perfectly in this scenario. It is assumed that VOs provide management logic to control the resources within the community.

External allocations may be needed due to spontaneous load surges or when resources cannot be provided by the VO. At a time, the VO A may require more resources than available to match the required quality of service. A VO monitoring service will trigger the buyer service (termed Prospector) to allocate the needed resources. The Prospector searches for markets that trade in the required resources and places a bid on them. Markets are services exposed by VOs that aim to trade in some of their resources. Seller agents (from B or C VOs for example) are triggered to sell resources when the monitoring service determines a surplus of resources within the VO according to some VO policies. Sellers create or place a bid on Markets depending on the suitability of the Market to trade in their resources. When a DyMRA component fails or is disconnected due to the inherent dynamism of VO members (i.e. someone switches off her computer) automatically and transparently to participants the service is reallocated to another suitable node within the VO. After the market clears and winning Seller services and Prospector services are notified, the resources are added to the Pool of external resources of the buying community.

DyMRA addresses this kind of scenarios by providing services to automatically allocate external resources into a VO. The main contributions of DyMRA are twofold; first, the components of DyMRA are deployed as services inside a VO and can, hence, be reallocated when its current location fails or disconnects, keeping the functionality available. The functionalities provided by DyMRA have been specially designed to enable the inter-VO resource allocation. That is, accounting services to monitor the amount of resources available and used within the VO, seller services to offer idle resources to third party VOs, buyer agent services to strategically bid in markets so as to obtain resources. Pool services to aggregate resources in a VO and many other required aspects related to the management and allocation of resources to attain the desired objective. Second, DyMRA proposes to distribute markets amongst virtual organizations, that is, enabling each VOs either to create a market as a service executed within the resources of the VO or to use an existing market running in the domain of another VO. Markets will be created on demand and following different strategies depending on the objectives of the VO. In this thesis strategies of buyer agents and seller agents have been considered out of the scope of the research due to the extension of the topic by itself, however it has been considered a next step and future work. On demand market creation places places our approach in a design space between a decentralized and a centralized architecture, which responds better to the dynamism of the targeted environment.

4.3.2 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 [61] which allows a small group of computers connected through Internet to participate in collaborative activities and sharing their resources (i.e. provides virtualisation of resources), while tolerating high levels of dynamism. This middleware also allows the deployment of services within a VO (or group) [56]. 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 re-instantiating 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. Markets are services running over LaCOLLA middleware so LaCOLLA acts as an overlay independent of the physical nodes forming the VO. Churn - the continuous process of node arrival and departure - may cause continuous reallocations of the services and make the system to work inefficiently. To cope with that problem, services and its state are replicated through the functionalities provided by the underlying overlay, i.e. LaCOLLA, keeping only one replica active in order to reduce the amount of resources required to keep the market running. Once a market fails or disconnects another replica restores the state from LaCOLLA repositories and becomes the active market.

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 work, 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.

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. 4.1), and how the posterior access to the traded resources is managed (fig. 4.2).

4.3.3 Trading process

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[2]. With this information, it contacts the

[2]The strategy taken to decide the price to bid for is out of the scope of our work, however is an

Prospector and asks it to acquire such resources.

2a. The Prospector looks for a suitable market according to the requirements from the accounting service 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.

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.

important aspect that will have important effects to the behaviour of the system. This work is considered a future line of research.

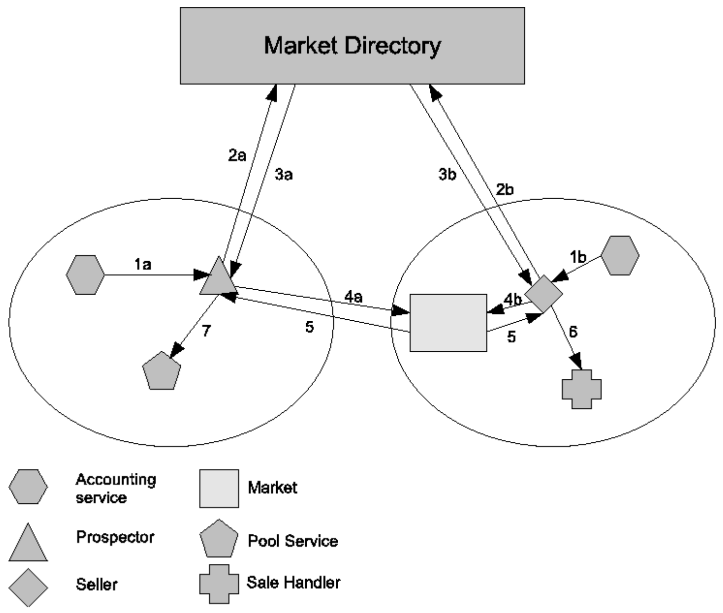


Figure 4.1: Interaction among components in the trading process.

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.

4.3.4 Accessing the resources

1. 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 case that external resources can be used, the client request is redirected to

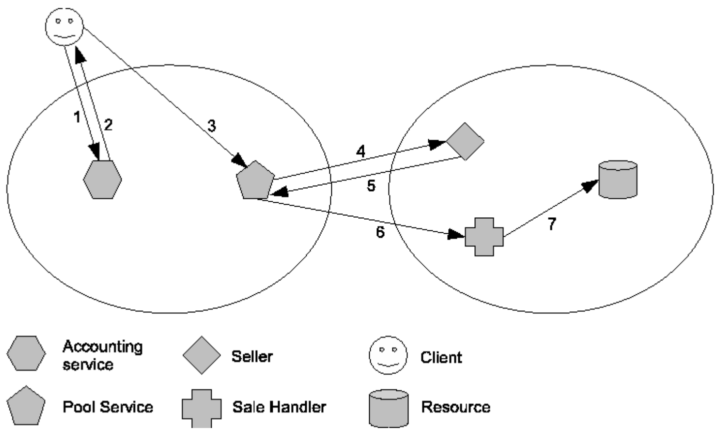


Figure 4.2: Interaction amongst components in the access process.

the Accounting service.

2. The Accounting service 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. In the former case, it tells the client which resource to use. Otherwise, it tells him to contact the Pool service.

3. The client contacts the Pool service, as if it was a local resource.

4. The Pool service chooses which of the external resources available to the VO should be used, and contacts its corresponding Seller. It sends the id of the sale it wants to use.

5. The Seller tells the Pool service the location of the Sale Handler that manages the specific agreement.

6. The Pool service contacts the Sale Handler, according to the conditions of the agreement, which may include, for example, symmetric key cryptography. It basically resends the request of the client.

7. The Sale Handler checks that the request of the Pool service does not violate the conditions of the agreement. After this, it uses the resources of the VO to fulfil the request of the Pool service.

As stated before, the services can change their locations due to failures or disconnections. This is not a problem inside a VO, as the system guarantees that clients, as well as other services, can contact any active service. To access external resources, though, the Pool must contact the Seller of another VO, whose location might have changed from the moment when the agreement was made. This can be solved in more than one way. A solution would be to use the Market Directory to store also the location of the Seller of each VO. This information would be maintained in a soft state, just like the one about markets, with the Sellers explicitly publishing their locations in the Directory. The Pool could then contact the MD to get the current location of the Seller, in case it cannot reach it in its previous location. This would solve the problem, but implies relying in an external entity (even though, as seen before, the MD can be implemented cooperatively by the VOs). A solution that only depends on the two VOs that need to communicate would be that both the Pool and the Seller keep the location of those Sellers and Pools, respectively, they have a deal with. In case one of these services is reallocated, it would notify all its "business partners" about its new location. Although it would be less probable, contact can still be lost if both Pool and Seller are reallocated at the same time. To further diminish this probability, these services could be replicated inside the VO. In the worst case, if all the replicas of both services fail together and the Pool of one VO can't contact the Seller of the other VO, the deal is broken, and both VOs will have to go back to the market.

4.4 Markets in DyMRA

Until now the main protocols and components used to mediate the inter-VO resource allocation have been presented. One of those components is the market. The market is a component that implements certain algorithms and functionalities to enable bidding, winner determination and settlement of agreements. Many of the current approaches such as PeerMart, Tycoon, Bellagio, etc. implement a single market mechanism (i.e a continuous double auction or a combinatorial auction) that make the approach restrictive in respect to the kind of allocations it can provide. That is, participants can only obtain the type of allocations that those specific markets can clear. For example a continuous double auction can only provide partial satisfaction of user requirements. i.e. even the user requires an specific set of resources such as a certain quantity of storage and a certain quantity of FLOPS the double auction won't ensure to obtain both at the same time.

For that reason a generic market component have been designed for DyMRA. The market component has been termed Configurable Auction Server (CAS) that provides a set of functionalities to control and configure the lifecycle of the market. The following sections present the main aspects of design and implementation of CAS.

4.5 Configurable Auction Server

Previous sections depicted the architecture of DyMRA, a resource allocation framework that provides functionalities to VO to access distributed markets through well defined protocols. In this section we focus on the core of the market server. That is, the component that holds the auction mechanism and provides specific properties to mange the lifecycle of the auction.

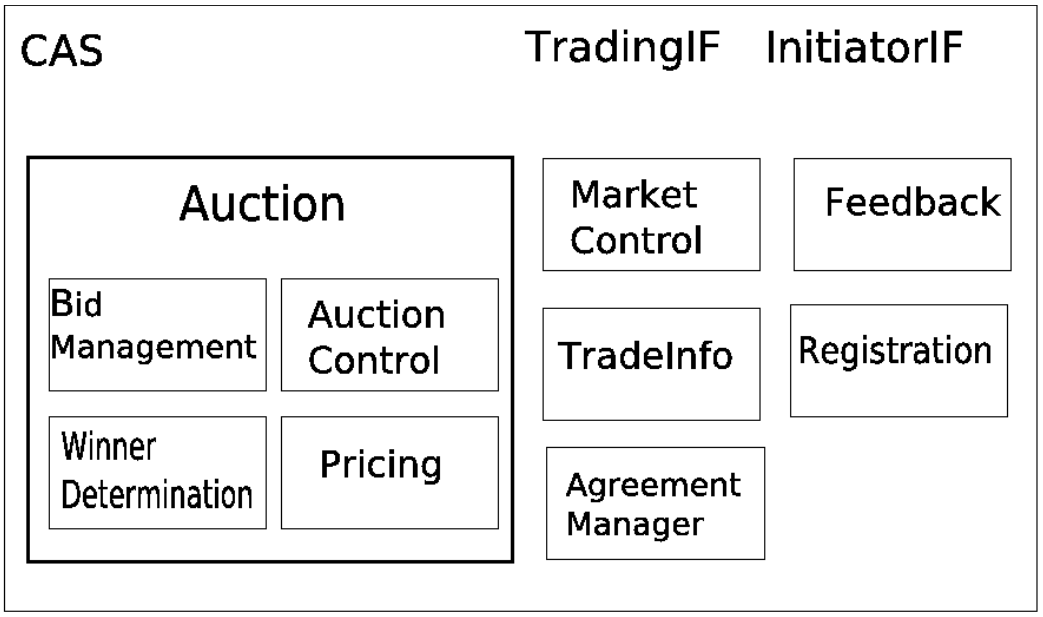


Figure 4.3: CAS Architecture.

4.5.1 Auction Server

We describe the Fractal [12] based Configurable Auction Server (CAS). An auction is a process that implements rules that govern registration, trading, determination of winners and prices. Based on established taxonomies [85], we propose an abstract approach to define components encapsulating such rules. Components can be combined to design markets that implement the rules governing a specific auction structure. A coherent set of components implementing a specific mechanism is described using the Fractal Architecture Description Language (see Figure 4.4 for a fractal component example).

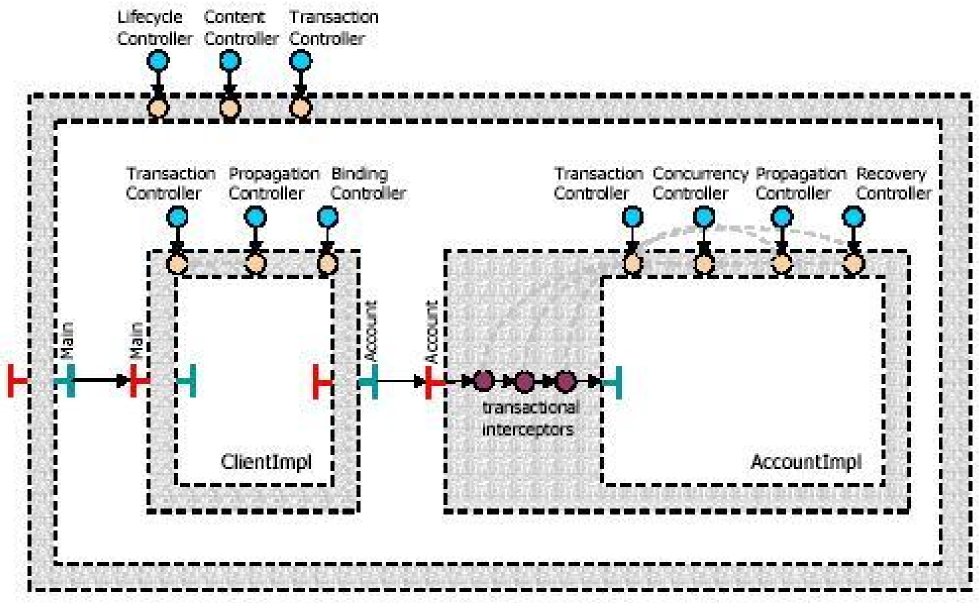


Figure 4.4: Fractal component example. Image from the ObjectWeb open source middleware at http://jotm.objectweb.org/jironde.htm.

This approach addresses:

- Configurability and administration: Market initiators can configure markets following their requirements, both at deployment and at run-time. Structural configurability such as addition/removal of components and modifying component interconnections and behavioural configurability such as attribute value setting and lifecylce modification is facilitated by Fractal.

- Reuse and extensions: Component based approach address re-usability and extensibility. The Configurable Auction Server facilitates rapid prototyping of complete new auction mechanisms or specific rules in an implemented mechanism. It separates generic system functionalities from trading specific functionalities.
- Deployment: Operating market-places with on-demand creation of auctions require powerful deployment mechanisms. The Architecture Description Language (ADL) and deployment tools from Fractal reduce the administration complexity (installation, configuration, tuning).

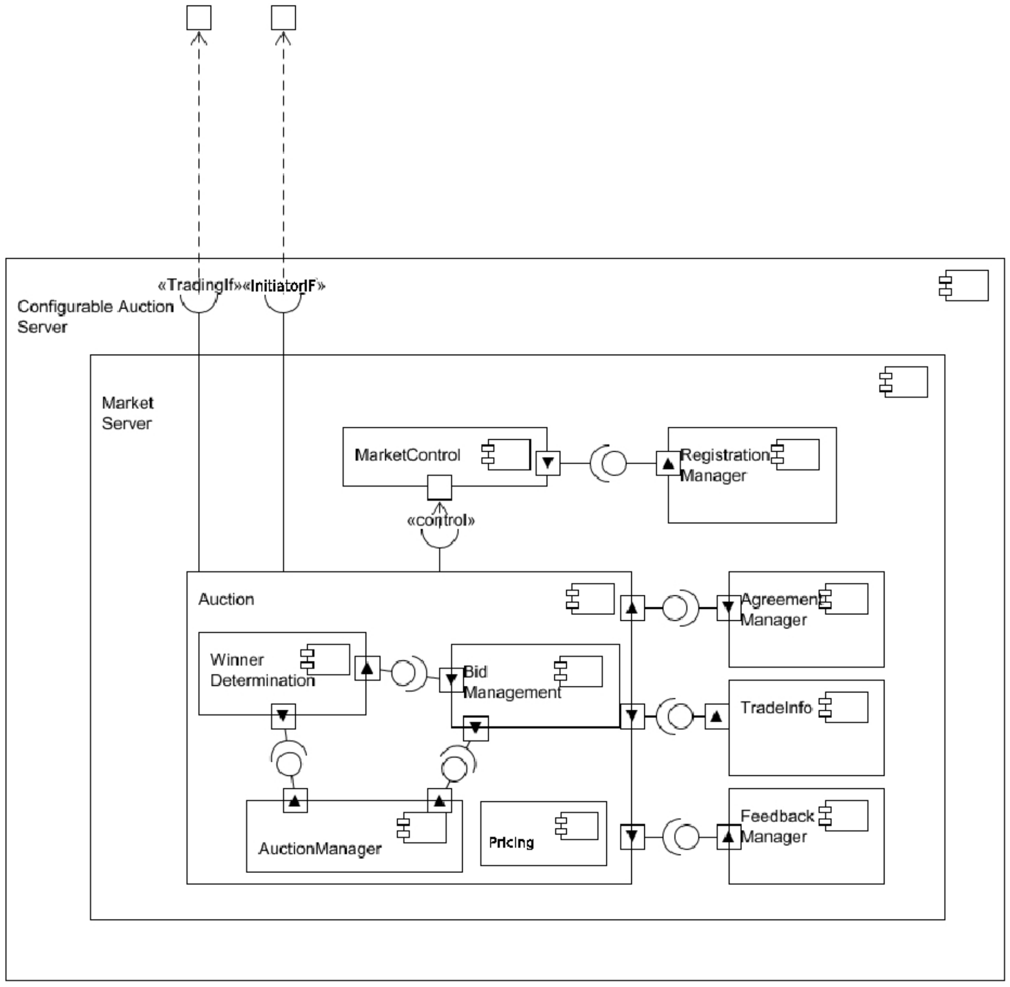


Figure 4.5: Component diagram of the Configurable Auction Server.

4.5.2 Architecture

The Configurable Auction Server (CAS) relies on the basic middleware services described in previous sections to provide abstractions and hide the heterogeneity and distribution of the underlying fabric [57, 60] (DyMRA and LaCOLLA in our implementation). It provides the market specific functionalities that enable the execution of auction mechanisms and the management and configuration of the lifecycle of the market. The CAS is, as state before, deployed as a service over LaCOLLA middleware and acts as a component of DyMRA.

In CAS, each component has a specified role within the framework (the market component), that is, corresponds to a market specific functionality such as the pricing algorithm or a generic system service such as authorization. The overall architecture is captured as a consistent combination of a subset of these basic independent components. It is obtained by assembling different components and binding their interfaces.

As can be seen in figure 4.5 the main components of the CAS are:

Market Server is the main container of auction and market functionalities. It governs the lifecycle of the market.

Auction Component is a composite component that implements the auction process and is composed of:

- The Bid Management that encapsulates bidding rules. Incoming bids are validated for conformance and either stored in waiting for clearance or dispatched to the clearing component. This composite offers interfaces that allow preprocessing of incoming bids to match the specific trading conditions of the market.
- Winner Determination component clears the auction. It is triggered by the Auction activity controller and matches the bids and offers stored by the Bid Management component.
- Pricing Component provides encapsulation for any specific pricing policy and determines final transaction prices.
- Auction Control component manages the lifecycle and maintains the state of the Auction component. It manages its functional sub-components and handles events.

TradeInfo Component encapsulates description of traded items and is configured at creation time. It's query interface informs participants of what is traded. Initiators configure the items to be traded at the market. The TradingIf is bound to this component in order to provide functionalities to query the resources offered by the market.

Registration Manager register authorized users.

Feedback Component interfaces allow participants to listen to market events such as current prices, termination, start of new round and final agreements.

Agreement Manager handles agreements and mediates the settlement phase.

Components are assembled to form markets. Specific rules e.g. a new pricing policy can be added to the platform by specializing the relevant component without perturbing the rest of the architecture.

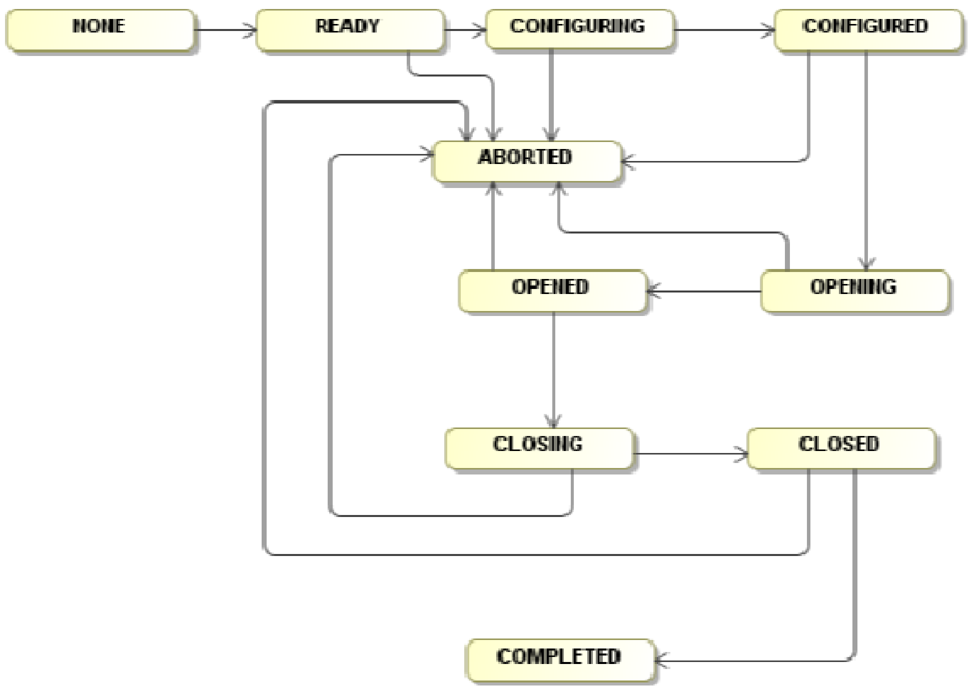


Figure 4.6: State diagram of the market server.

4.5.3 Workflow and Control

The independent components need to be controlled and orchestrated during the life-time of the auction. Auction activities are in general triggered by events such as timers and method calls. The CAS is designed as an asynchronous event-based framework to co-ordinate the activity of the participating components. Components are organized in a tree-like hierarchy with a manager associated at each node in the hierarchy. The auction server has a set of global states inherited by all its sub-components and each sub-component may have its private sub-states. Managers at each level trigger transition from states and implement synchronization points.

We have identified two generic workflows to support a wide range of auction mechanisms. The single shot market workflow and the iterative market workflow. The choice has been taken after an analysis of market mechanisms such as English Auctions, Vickrey Auctions, Dutch Auctions, Double Auctions and Combinatorial Auctions. For those institutions single shot and iterative forms exist. Single shot markets are those that after a bidding phase the market clears and the auction terminates. In iterative markets in contrast, bids can be re-submitted after a clearing following certain rules. Termination is triggered by different rules. Single shot market workflow implements the states depicted in figure 4.7. As can be seen, the market can only move to closing state after the clearing has been done. Iterative markets in contrast can start a new bidding phase after the clearing ends. Workflows can be added programmatically as managers implementing the main APIs provided by the CAS.

Functional behaviour

CAS has two type of components: passive and active. Primitive components are typically passive, they mainly respond to method invocations. Active components have their own thread and active composite components control the lifecycle of their children. Such managers also maintain timers for their sub-components. Timers are configured by auction initiators as part of the initialization phase. At the root-level, the CAS is managed by the MarketServer component (see figure 4.5), that controls the Auction composite, the RegistrationManager, the AgreementManager, the TradeInfo component and the FeedbackManager. The AuctionManager, i.e. the controller of the Auction composite manage the states of its sub-components. State transitions for the Auction component can be seen in figure 4.7 either when timers expire or when controlled components end some action.

Passive components are controlled by their parent manager and implement the methods of the management interface.

4.5.4 K-DA Mechanism

The K-pricing based double auction is a simple yet powerful mechanism to trade multiple units of single items; an item could be CPU or Storage. This section describes how the implementation of this mechanism fits into the previously described architecture and the specific extensions to this well known mechanism to trade time-differentiated resources that will be presented in the following chapter. As it will be seen, the k-DA mechanism is extended to lease computational resources at distinct times in units called time-slots. It provides tools to decompose bids with imprecise time-specifications and adjusts the demand across the different time-slots. Besides, we developed one algorithm[3] to clear multi-unit k-double auctions when bidders formulate substitute bids and complementary bids. Bids will be organized in lanes, where each lane is responsible for one type of item. In our case, the time-slot number distinguishes one item from another. For each lane, the clearing algorithm will determine the winners for that lane. The main issues to be tackled are:

- Order of clearance of lanes. Lane ordering is irrelevant in the case where there are no substitutes, i.e., every user precisely specifies the required time-slot. XOR Bids with substitute time-slots create dependencies between lanes. In this case the order of clearing of lanes affects the overall efficiency.

- Remove constraint on co-location of lanes. Executing lanes on more than one physical machine may improve overall performance, i.e., reduce clearing time.

[3]Multi-Lane Double Auction (MLDA)

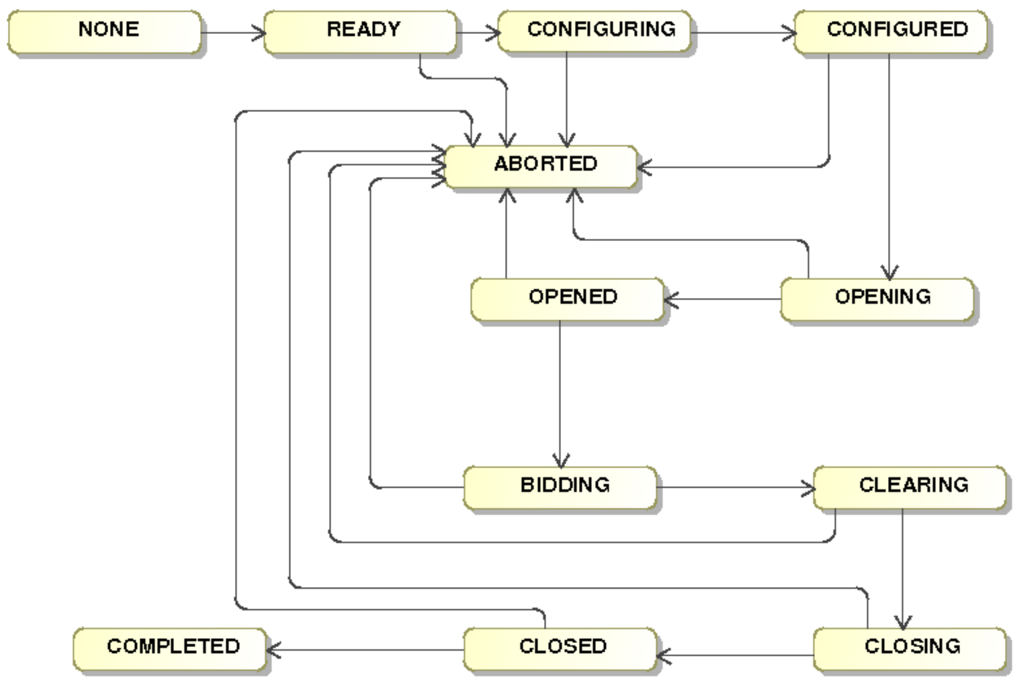


Figure 4.7: State diagram of the single shot auction.

4.5.5 Deployment, Configuration and Execution

The CAS is designed using Fractal and its architecture is described using the Architecture Description Language. The ADL describes the architecture, i.e., the set of concrete components and the binding between the components. Using ADL, we decouple the functional program development from management tasks such as deployment and life-cycle management. Besides, Fractal [12] enables dynamic binding and deployment of components through lifecycle and binding controllers that can be accessed through the InitiatorIf API offered by the CAS. Therefore, CAS can be configured statically by means of a set of ADL configuration files but it also can be dynamically configured at runtime.

In the CAS several Fractal controllers were used in the design: The Attribute controller is used for configuration of the market components, the Life-cycle controller to hierarchically start/stop components, the Content controller to add/remove content to the sub-components and the Binding controller to dynamically bind components.

ADLs are specified at programming time and are used to deploy market instances. Run-time tools parse the declarative description of the architecture, deploy the application components and establish their bindings. Components may be co-located on one machine or distributed on multiple nodes without requiring modifications on the application logic. Components interact as if they were local to each other even they are running in different nodes. This functionality is transparently provided by Fractal. Dynamic binding of components also enable self-management; the auction server can be reconfigured on events such as failures, increase in load. For instance, new lanes can be dynamically instantiated based on monitoring of current load, where load is measured in number of bids and asks.
Once deployed, the InitiatorInterface API is used to configure at run-time; timers, participation constraints, bidding constraints and the traded items.

4.6 Implementation

This section describes the Java implementation of DyMRA and their market holders CAS. The implemented prototype of the proposed architecture is composed of the Prospector, Seller, Pool, SaleHandler and the Market components. They have been implemented as deployable services over the LaCOLLA middleware. The Market itself have also been developed as a composite Fractal component that exposes a generic API that allow different mechanisms to be implemented. The Market held two market mechanisms, namely it implemented two variants of the double auction [10] protocol which enables buyers and sellers to submit bids for multiple units of a single resource (i.e storage capacity, cpu capacity and applications).

The first variant was the k-DA mechanism which provides a specific implementation of the BidCatalog component. In our case we developed an engine based on the four heap algorithm described in [90]. This algorithm have been chosen because is one of the most used algorithm for the allocation of resources in computational environments. For example, Tycoon [52], PeerMart [44] and the JASA framework [71] make use of it. The FourHeapLaneEngine uses four heap data structures to maintain the winning bids and winning asks respectively. The second variant is the Multi-Lane Double Auction that will be presented in the next Chapter. In the MLDA the received bids are processed and organized in multiple lanes, each corresponding to disjoint sets of time-slots. Each lane of the MultiLaneBidCatalog is an auction engine.

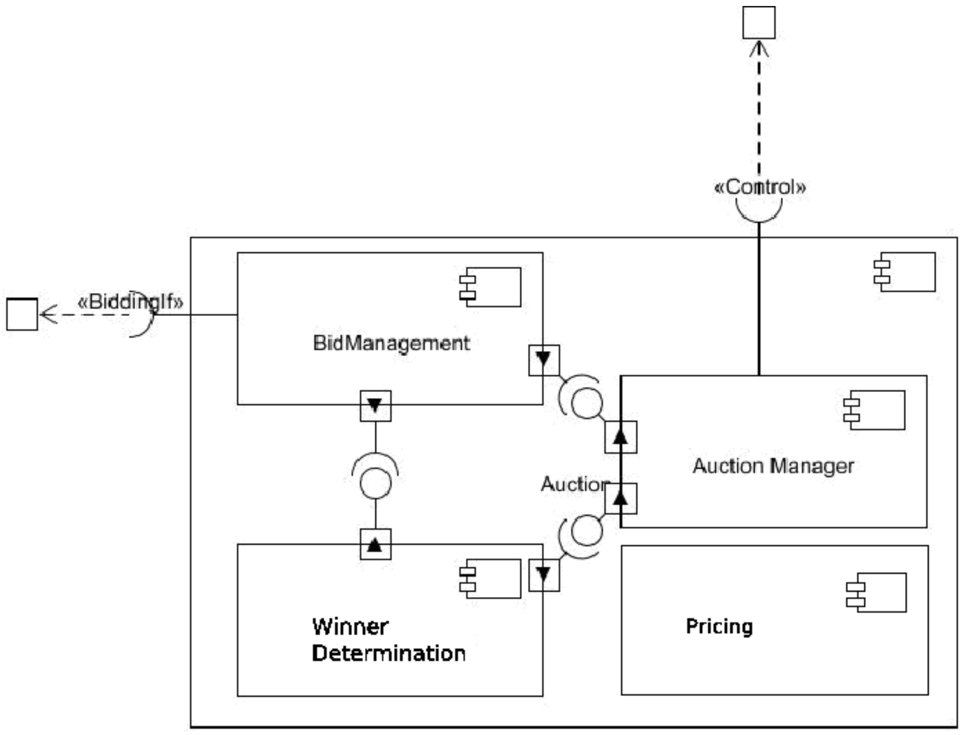


Figure 4.8: Sub-components of the Auction composite component.

The MarketDirectory has been implemented as a centralized index, but, as mentioned above, it can be easily substituted with a decentralized approach [17, 41]. 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. In the context of our evaluation availability refers to the fact that when a buyer or a seller aims to place a bid in a market, the market is active and can accept the bid. We want to prove that markets can be reallocate transparently in the presence of failures and the reallocation process can be considered to be transparent.

We expected reallocation times of the order of milliseconds depending on the available broadband. The time for reallocation depends merely on a set of messages exchange amongst several components of the underlying middleware and the time taken for the service to recover the state.

To corroborate our hypothesis 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 MarketDirectory 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. For simplicity, Markets have been assigned a limited lifespan, after which they resolve the auction and send the results to the clients. This implies that, periodically, a Prospector or a Seller will have to create a Market, thus decreasing the perceived availability. Markets could also be permanently active, which would increase the availability of the system. There is, though, a trade off between the obtained availability and the resources spent to keep the market active.

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 which corresponds to 1,8 failures per hour, and a probability of ordered disconnection of 0,0025 that corresponds to 9 disconnections per hour. In the more dynamic of the two (G2), the probability of failure per iteration was 0,005 (9 failures per hour), while the probability of disconnection was 0,008 (28,8 disconnections per hour). Tests lasted 500 iterations. This parameters were chosen according to the experiments realized in LaCOLLA [60] to validate their functioning. In LaCOLLA different groups were characterized and the system validated according to the behaviour of different groups of participants. DyMRA have been designed to expand group capacity through the allocation of resources provided by other groups. These groups are formed of user's collaborating to achieve a common objective as in the case of LaCOLLA and consequently we considered that they can be characterized according to the experiments in LaCOLLA.

The data we analyse is the number of bids that arrive to the market, in relation to the number of bids issued by the group because this metric represents the times that the market have been available. 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 because markets have a limited lifespan and are created on demand, which 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, only that it will have a bigger delay.

Fig. 4.9 shows the availability (percentage of successfully issued bids) obtained in 20 executions, for both G1 and G2. We see that, as expected, the availability is higher in G1, decreasing in G2 because of the higher level of dynamism.

Figures 4.10 and 4.11 show the cumulative distribution function for both G1 and G2. For G1, 50% of the executions obtain an availability of 70% 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.

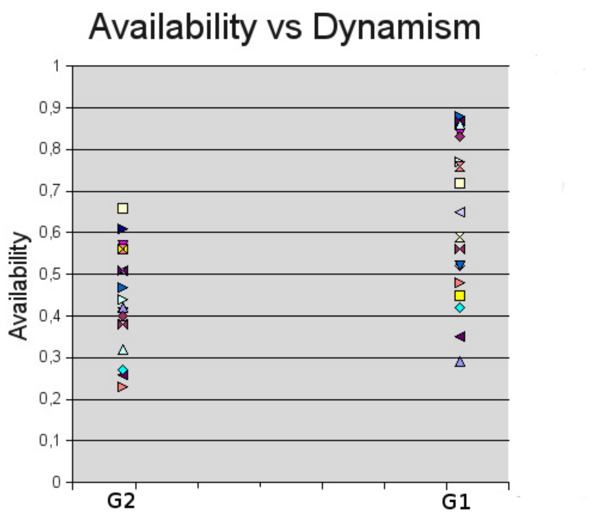


Figure 4.9: Availability vs level of Dynamism.

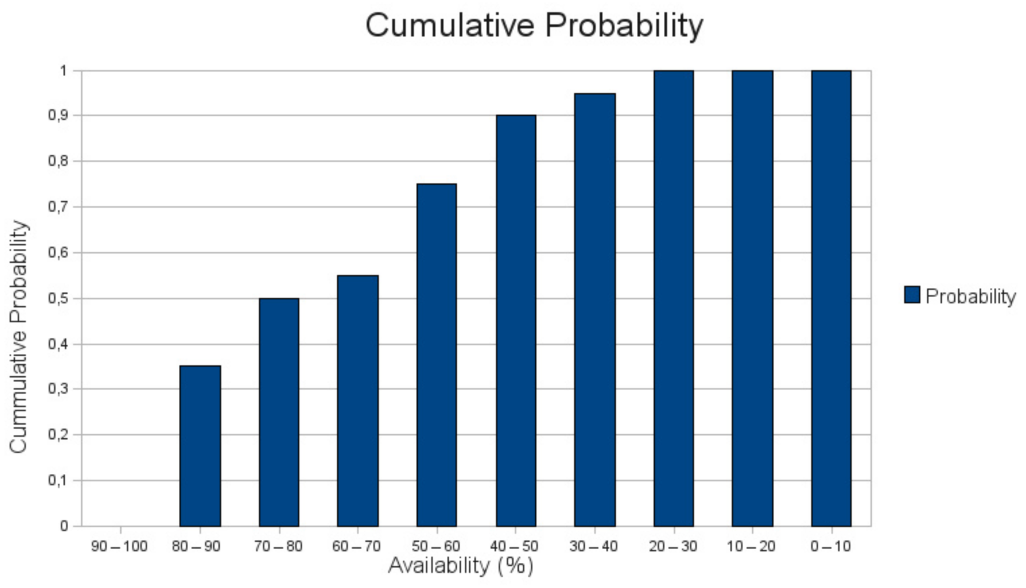


Figure 4.10: Cumulative probability of availability levels for G1.

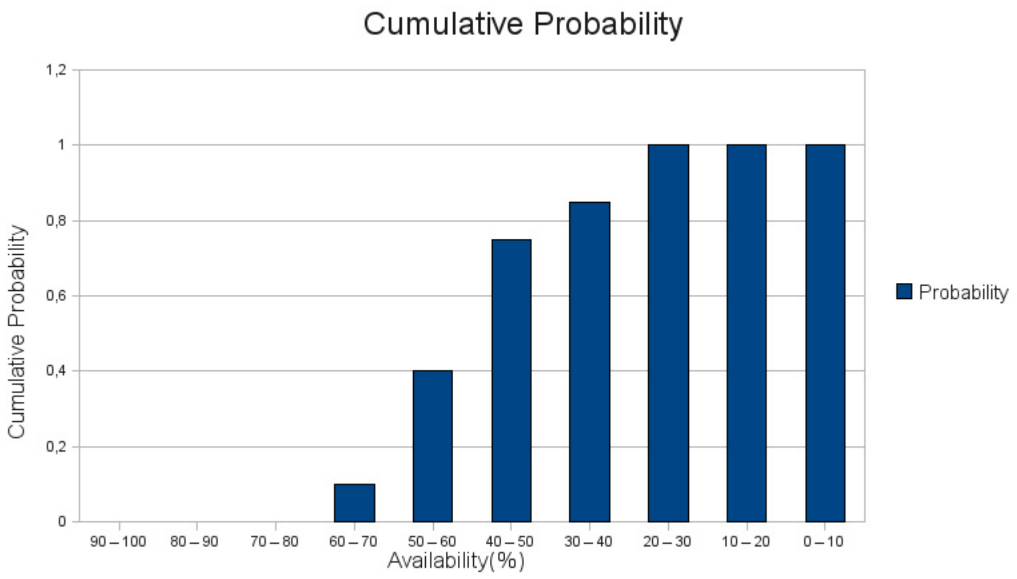


Figure 4.11: Cumulative probability of availability levels for G2.

4.7 Conclusions

The chapter presented an architectural approach to decentralized resource allocation in Contributory Systems. DyMRA, a peer-to-peer based architecture has been proposed as a brokering infrastructure to negotiate for external resources. Prospector and Seller agents act as autonomic components that decide based on VO policies to acquire resources in external markets. Both Seller and Prospector have the ability to create markets when no markets for an specific resource exists. VO resources are pooled and controlled by the Pool Service component and internally accounted by the VO Accounting Service. Seller components are in charge of VO's idle resources, Sale Handlers control and monitor the external access to resources acting as mediators components. The market component has been developed as a separate and configurable component following the separation of concerns paradigm. Market components have been divided by functionality and generic aspects have been separated from market specific aspects. A Fractal-based component approach have been used to provide flexible and easy component assembling which provides non-functional properties such as modularity, configurability and decentralization. The architecture presented in this chapter aimed to address the properties of Contributory

Systems, characterized by dynamism and evolution which makes monolithic approaches infeasible. Genereicity, re-usability, flexible deployment and configuration as well as high levels of autonomism and self-reallocation constitute the main contributions of DyMRA and CAS. These properties have been addressed so as to obtain an architecture suited for distributed and dynamic environments such as Contributory Systems. This work helped us to understand the nature of ad-hoc infrastructures for collaborative computing in either of its variants. It is expected that some of the concepts depicted in this section such as autonomous and transparent reallocation of markets, on-demand creation of markets to address decentralization and genereicity and re-usability of components will be taken in consideration in the next generation of Utility Computing infrastructures that will bring unlimited computational capacity to the edges of Internet as it is aimed in the Grid4All European project [25] where part of this work is being carried out.

Chapter 5

Multi Lane Double Auction

5.1 Introduction

Previous chapters of this thesis presented our efforts to build a decentralized resource allocation framework adapted to dynamic environments such as Open Grids and Contributory Systems. The introduction of the thesis has characterized these scenarios and presented the main issues that resource allocation frameworks need to deal with. In this chapter we focus on the most economic part of the thesis and presents our approach to develop a market mechanism specially adapted to trade computational resources.

The motivation for that approach is to facilitate the allocation of time-differentiated resources in computational environments. Grid resources are always accessed during limited periods of time that means that their allocation is always through leases. Current existing auctions do not provide support to allocate time-differentitated resources considering them differentiated items. In this section an introduction to the market mechanism spectrum is presented so as to orientate readers and to position our contribution.

Market mechanisms have been extensively studied and there are multiple approaches used in the context of the allocation of computational resources. Market mechanisms are processes for determining trade prices, i.e. how bids and asks can be exchanged in the market to determine a trade. Market mechanisms are at the centre of market-based allocation framework. The choice of market mechanisms can significantly affect the market efficiency, the complexity (in terms of both participating and implementation) and the way in which resources can be scheduled in Grid computing environments.

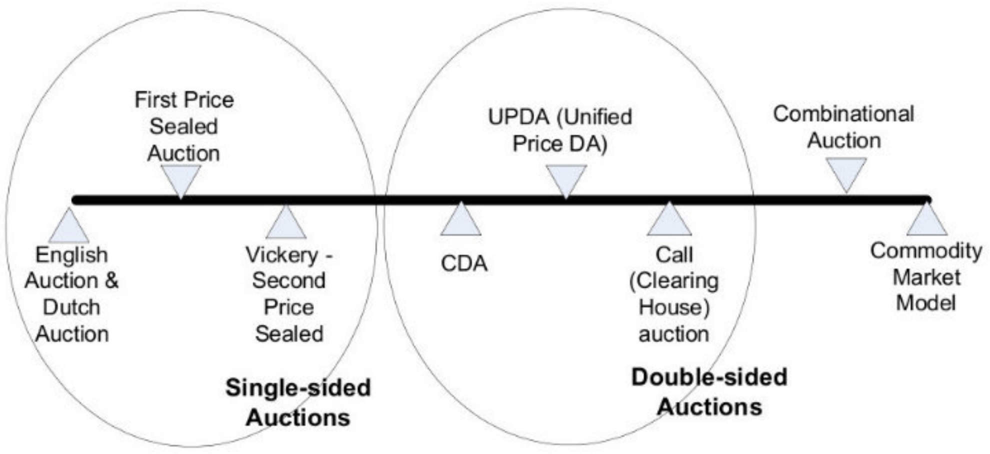

First Price Sealed Auction
UPDA (Unified Price DA)
Combinational Auction
English Auction & Dutch Auction
Vickery - Second Price Sealed
CDA
Call (Clearing House) auction
Commodity Market Model
Single-sided Auctions
Double-sided Auctions

Figure 5.1: Spectrum of Market formulations.

Figure 5.1 presents a spectrum of the main market models that exist. The commodity market model is based around centralized information and strives to generate a price that ensures the equilibrium of supply and demand. The same, or similar, resources are traded identically in a commodity market. A consumer does not purchase a specific resource, but rather takes one of many equivalents. At the other end of the spectrum, the single-sided auction is more dynamic and flexible, as negotiation can be done for every individual resource, but at the expense of efficiency. Resources are allocated on an individual basis and single-sided auctions are easy to implement and straightforward to participate in. Different market formulations in the spectrum will be discussed.

Single-sided auctions are the traditional and most cited forms of auction: an institution with an explicit set of rules determines resource allocation and prices on the basis of bids from the market participants [63].

As its name suggests, a single-sided auction is a mechanism for one-to-many price negotiation, i.e. the competition comes only from one side (either consumers or providers). William Vickrey established the basic taxonomy of single-sided auctions based upon the order in which prices are quoted and the manner in which bids are tendered. He established four major types of single-sided auction [88], as described below.

- English auction, also known as an open ascending price auction. This type of auction is arguably the most common form of auction in use today. Participants bid openly against one another, with each subsequent bid higher than the previous bid [63]. An auctioneer may announce prices, bidders may call out their bids themselves (or have a proxy call out a bid on their behalf). In some cases a maximum bid might be left with the auctioneer, who may bid on behalf of the bidder according to the bidder's instructions. The auction ends when no participant is willing to bid further, at which point the highest bidder pays their bid. Alternatively, if the seller has set a minimum sale price in advance (the 'reserve' price) and the final bid does not reach that price the item remains unsold. Sometimes the auctioneer sets a minimum amount by which the next bid must exceed the current highest bid, termed price increment rule. The most significant distinguishing factor of this auction type is that the current highest bid is always available to potential bidders. A computerized English auction normally specifies a hard deadline for implementation reasons. The main problem with the English auction is the winners curse a bidding race amongst bidders results in the bidder who wins the auction being the bidder who has most likely over-valued the resource.

- Dutch auction also known as an open descending price auction [54]. In the traditional Dutch auction the auctioneer begins with a high asking price which is lowered until some participant is willing to accept the auctioneer's price [62]. The winning participant pays the last announced price. The Dutch auction is named for its best known example, the Dutch tulip auctions. ("Dutch auction" is also sometimes used to describe online auctions where several identical goods are sold simultaneously to an equal number of high bidders.) In addition to cut flower sales in the Netherlands, Dutch auctions have also been used for perishable commodities such as fish and tobacco [62]. In practice, however, the Dutch auction is not widely used.

- Sealed first-price auction, also known as a first-price sealed-bid auction (FPSB). In this type of auction all bidders simultaneously submit sealed bids so that no bidder knows the bid of any other participant. The highest bidder pays the price they submitted. [54, 62] This type of auction is distinct from the English auction, in that bidders can only submit one bid each. Furthermore, as bidders cannot see the bids of other participants they cannot adjust their own bids accordingly. Sealed first-price auctions are commonly used in tendering, particularly for government contracts and auctions for mining leases.

- Vickrey auction, also known as a sealed-bid second-price auction [54].This is identical to the sealed first-price auction except that the winning bidder pays the second highest bid rather than their own. This is very similar to the proxy bidding system used by eBay, [31] where the winner pays the second highest bid plus a bidding increment (e.g., 10%) [54]. Although extremely important in auction theory, in practice Vickrey auctions are rarely used [62]. The most important problem with Vickrey auction is bid shading, that is, placing a bid which is below the bidder's actual value for the item. Such a strategy risks losing the auction, but has the possibility of winning at a low price.

Double-sided auctions allow many-to-many price negotiation. In a double-sided auction, both buyers and sellers can submit bids and asks for standardized units of well-defined commodities and securities [37]. The main institutions to be considered are:

- Single item double auctions are auctions that simultaneously mediate among multiple buyers and multiple sellers There exist two main institutions for double auctions. (i) The clearing house or call auction, which clears periodically and (ii) the continuous double auction (CDA). The CDA matches buyers and sellers continuously as bids arrive. For homogeneous items, a simple CDA implementation may maintain a queue of bids sorted in increasing order of price and a queue of offers in decreasing order of price. If an incoming bid is greater than the head of the offer queue it is matched to that offer, or otherwise inserted in the bid queue. There are different strategies to set the price for that match. The k-double auction uses a parameter $k(0, 1)$ that determines how trades are priced. In that case the transaction price is set at :

$$price = k \times \ bidPrice + (1 - k) \times \ askPrice \tag{5.1}$$

The Mth and (M+1)st double auction computes the Mth and (M+1)st prices where M is the number of sell bids. Such prices may be computed by sorting the bids in descending order and identifying the Mth and (M+1)st elements in the list. The prices between Mth and (M+1)st bids inclusively represent the range of prices for which supply balances demand. Vickrey [88] demonstrates that the M+1 double auction is incentive compatible for buyers and M double auction is incentive compatible for sellers.

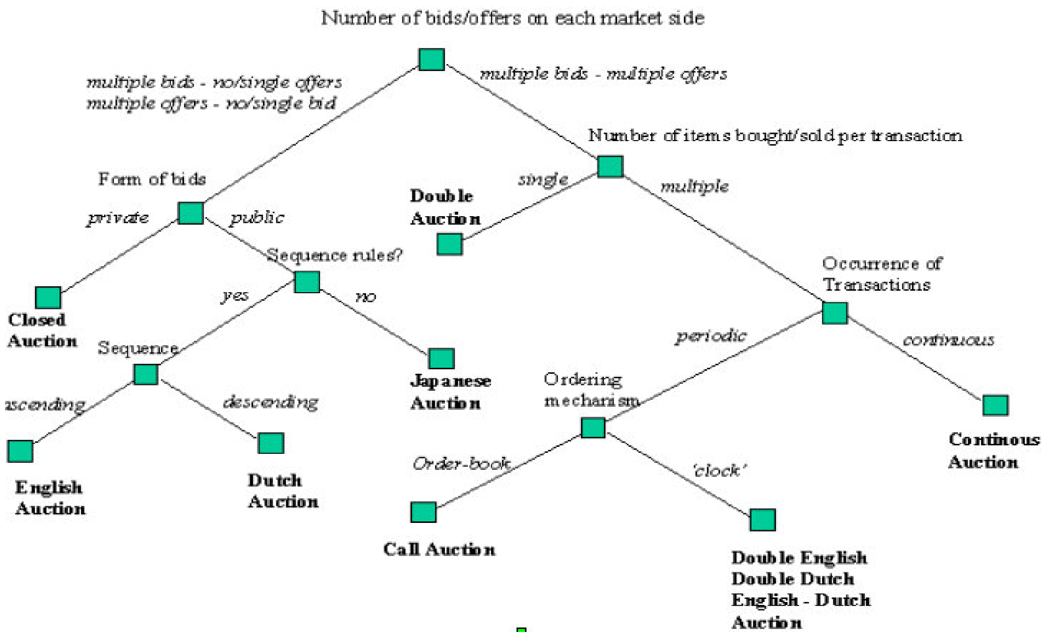


Figure 5.2: Auctions classified according to different properties.

- Combinatorial auction [65, 79, 91] is an auction where bidders can bid (sell) entire bundles, that is, combinations of items of goods. This has the advantage of eliminating the risk of a bidder of not being able to obtain complementary (sum of valuations of items is less than the valuation of the entire bundle) items. Market participants express their preferences as bundles of resources that need to be matched. CA is computationally complex. Formally the combinatorial auction problem can be expressed as a special variant of the weighted set packing problem (WSPP) [26]. The winner determination problem is to label bids as winning or losing so as to maximize the auctioneer's revenue (under the constraint that each item can be allocated to at most one bidder). This NP-complete problem can be solved with dynamic programming techniques that are time intensive. With restrictions on the combinations of bids, winners can be determined in polynomial time this gives rise to economic inefficiencies since bidders may not be able bid on their preferred combinations. Search algorithms have been investigated to find revenue maximizing allocations these use bid ordering heuristics, such as ordering bids by their contribution to revenue. A typical bid ordering criteria is the average price per good.

5.2 Requirements

As discussed in previous chapters, our objective is to enable on-demand delivery of resources to virtual organizations as they where simple utilities. Implementation of resource market places in open environments must take into account major non-functional requirements described below.

Scale: considering a population in the scale of millions of users sharing their resources across the Internet either on a for-profit or a non-profit basis becomes a challenge in handling the complexity that derives from the quantity of resources, users and numbers of concurrent actions (for example requests for resources).

Heterogeneity: Open systems consist of a diverse range of resources (for example different processor architectures, speeds, middleware etc.), diverse range of applications and application demand functions (such as elastic demand for resources, real-time), diverse usage patterns, resource owner behaviours (for example may decide to trade resources at different times), and consumer endowments.

Dynamicity: Ad-hoc Grids and Contributory systems shall support the on-the-fly creation of highly dynamic virtual organizations where participants and resources can join and leave at will. The lifetime of a Grid service or collaboration might range from a few hours (e.g. an e-learning interactive session) to several years (e.g. twinned schools cooperating on an educational project). The central issue in resource allocation is the nature of resources itself which has been discussed in Chapter 3.

The auction spectrum gives us the map of current existing mechanisms that can be used to allocate resources in Grid. However, we wonder about whether a single market mechanism can be sufficient to deal with the set of requirements raised from our objective. It seems reasonable to assume that Contributory systems require multiple market mechanisms cohabiting so as to handle heterogeneity, dynamics and constant evolution. It is clear that in such environments buyer agents do not require the same type of resources and resource providers do not offer the same type of resources.

Even in the case that multiple market mechanisms are supported, we wonder about the suitability of current auctions to allocate efficiently time-differentiated resources (usually provided by many different resource providers) such as most of the resources in a Grid, for example Combinatorial Auctions enable the trade of time-differentiated resources but at a high computational cost. Other simpler auctions does not enable multi-item trading but only one type of item can be handled. Straightforward approaches may require multiple instances of single item auctions to manage time-differentiated resources.

5.3 Objective

As a part of our work we aimed to develop a double auction institution that can be used to allocate time-differentiated resources. The main reason to develop a time-differentiated item auction is that, until now, time-differentiated resources were either traded by computationally costly mechanisms such as Combinatorial Auctions or by multiple instances of single-item Double Auctions. We envisage that by developing a time-differentiated item auction we can improve the obtained social welfare regardless of multiple instances of single item double auctions in the presence of substitute preferences. Besides, computational efficiency can be improved in respect to multiple instances of single item double auctions because the overhead of maintaining multiple instances can be reduced.

As a result of the research presented in this chapter, the Multi-Lane Double Auction (MLDA) has been developed. The MLDA can be set in between the computationally hard approach of the Combinatorial Auction and the restrictive approach of the Double Auction.

The following example will be used to illustrate the need for the MLDA:

Lets consider a buyer willing to allocate computational capacity for a certain time. Imagine that the computation he aims to do requires at least certain capacity to end in an strict deadline. However, the utility of the buyer increases as early the application finishes (see Figure 5.3). Thus, the buyer has some interest on obtaining more resources than the strictly needed in order to finish his computation sooner. In a computational environment with multiple mechanisms cohabiting the bidder can split his bid in two parts. One strict part will require the minimum quantity of resources so that the computation can be finished within the strict deadline. This bid will be submitted to a combinatorial auction that provides complete allocations. Once upon the buyer obtains such quantity of resources, he can formulate another bid with imprecise requirements, this means that the buyer wants more computational resources but he doesn't care when within the deadline. Any resource that can be obtained will improve his utility. In this case, the buyer has to formulate a bid for several units of computational capacity between the start of the computation and the strict deadline. The bid does not require complete satisfaction and allocation obtained will improve the utility of the buyer.

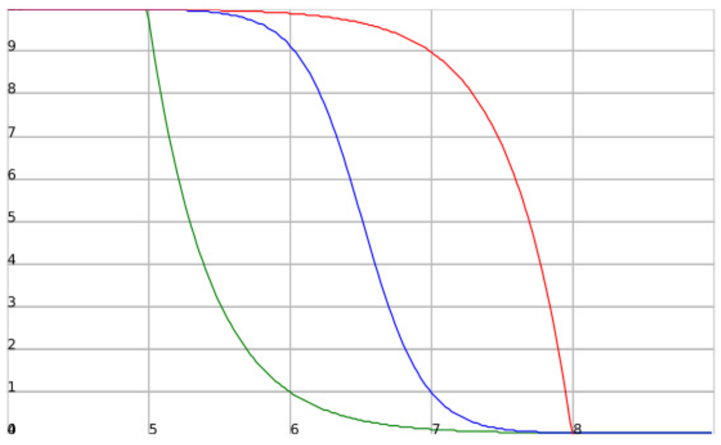


Figure 5.3: Possible valuation functions of the job w. r. t. completion time.

In a previous chapter, a bidding specification has been presented. The bidding specification was specially designed to formulate bids for computational resources in an environment characterized by multiple market mechanisms cohabiting. One of the main tools provided as a complement to the bidding specification enables bids to be decomposed into time-differentiated bids following a given pattern. The decomposition turns out to be useful when bidders have imprecise requirements for certain time-slots. For example, a bid B1 described as follow:

B1=item=1GHz CPU, qtt=2, start time=9:00, end time=14:00 where time-slot is of 1h, partial satisfaction is allowed

B1 requires at most 2 CPUs of 1GHz for one time-slot each one between 9:00 and 14:00. Given that requirement, the utility perceived by the bidder obtaining any CPU of 1GHz between 9:00 and 14:00 can be considered the same. Thus, the bidder is indifferent of any of the time slots he receives. Amongst multiple possible decompositions, the bid can be expressed as an OR bid for the same resource but for any of its different time-slots of certain granularity.

Using the current double auction approach, this bid can only be placed into a double auction selling CPUs of 1GHz between 9:00 and 14:00 as a whole, so that , the bid will be considered for 2 allocations of 1GHz CPU between 9:00 and 14:00 that obviously is not what the bidder requires. Besides, if the bid wants to be placed into a double auction selling items as units of one time slot, the bid requires some pre-process in order to transform it into an explicit formulation such as:

B1'= (B2 OR B3) XOR (B3 OR B4) XOR (B4 OR B5) XOR (B5 OR B6)

where:
B2=item=1GHz CPU, qtt=1, start time=9:00, end time=10:00 where time-slot is of 1h, partial satisfaction is allowed
B3=item=1GHz CPU, qtt=1, start time=10:00, end time=11:00 where time-slot is of 1h, partial satisfaction is allowed
B4=item=1GHz CPU, qtt=1, start time=11:00, end time=12:00 where time-slot is of 1h, partial satisfaction is allowed
B5=item=1GHz CPU, qtt=1, start time=12:00, end time=13:00 where time-slot is of 1h, partial satisfaction is allowed
B6=item=1GHz CPU, qtt=1, start time=13:00, end time=14:00 where time-slot is of 1h, partial satisfaction is allowed

Given the bid B1', an auction mechanism able to support XOR bids for multiple time slots is required. Current double auction approaches will require multiple instances of an auction to deal with B1'. As state before, multiple instances require a higher levels of management and coordination and introduce an administration and computation overhead in the environment making the approach more inefficient and costly. To cope with that problem, we propose to extend the current double auction mechanism so that it can deal with multiple items at the same instance.

In contrast, bids for precise items can simply be formulated as:

B7=item=1GHz CPU, qtt=1, start time=9:00, end time=10:00 where time-slot is of 1h, partial satisfaction is allowed

Given that brief remainder to the way bids are formulated and its decomposition when bidders are imprecise on their requirements, we can introduce the MLDA structure and algorithm.

5.4 The Multi-Lane Double Auction

This section describes the data structures and methods of the multi-lane double auction. The idea behind this approach is to split the item that is offered into its time-differentiated units. For example if the item is offered in multiple time slots, and the time slot size is of one hour, the auction will be split in as many lanes as time slots are offered. This approach can also be taken by other metrics out of time slots. For example, for storage capacity, the available space can be sold in units of a fixed size (e.g., 100 Mb), creating as many lanes as units are offered. In the remaining of the discussion we will illustrate all the examples considering the case for time-slots, however, any other case can directly be mapped to this one.

A lane represents an item as it would be sold in any single item double auction. Bids and asks will be introduced into lanes corresponding to the item they are representing. The number of lanes in an auction will be obtained once the auction is configured. At initial offer setting, the traded item will describe the time range for which the item is offered. The time range will be split in as many time slots as the time range divided by the time slot granularity indicates. The number of time slots obtained will directly determine the number of lanes for that auction. For example, given an offer description to create a double auction such that:

S1=item=1GHz CPU, start time=9:00, end time=14:00 where time-slot is of 1h

Five lanes can be created each one representing one time slot. As a generalization, the number of lanes can be obtained by:

$$L = E - S/U \tag{5.2}$$

where S is an offer such that $S = R, Q, S, E, U$ and where:
R is the resource description.
Q is the quantity of the resource.
S represents the start time.
E represents the end time.
U represents the unit size.

5.4.1 Preciseness vs Impreciseness

By precise bids we understand those bids that are specific for one time slot and do not express any substitutable preference. Contrarily, imprecise bids are those that can be introduced in more than one lane because the bidder has substitute preferences for any of the items in its bid. Current double auctions does not deal with impreciseness requiring bidders to formulate precise bids. Impreciseness enhances the flexibility of the market place and facilitates bidding and preference elicitation since buyer agents do not need to specify precisely the overall set of possible substitutes he will accept.

5.4.2 MLDA structure and general operations

The auction can be constructed as set of lanes where each lane contains the data structures used to maintain bids an asks.

The main operations offered by the multi-lane double auction 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 bid 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 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.

5.4.3 How does a double auction work?

Double auctions determine the winners set amongst multiple buyers and sellers, each offering to buy or sell a single items, to clear the auction, the M th and (M + 1)st prices are computed, where M is the number of sell bids. We assume 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 M th 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. Specifically, if pM is the M th-price, and pM +1 is the (M + 1)st-price, the k-double auction will set:

$$p = k \times (pM + 1) + (1k) \times pM$$

$$s.t. 0 \leq k \leq 1$$

Furthermore, the M th and (M + 1)st prices delineate the set of currently winning bids, which we refer 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.
The procedure can be extended to multi-unit, partially satisfiable bids simply by treating each unit offered as a separate bid. In this case, M is the number of units offered for sale, and N is the total number of units in all bids. The M th-price ((M + 1)st-price) is set by the price on the M th ((M + 1)st) highest unit.

5.4.4 MLDA operations

The MLDA algorithms have been developed following the same idea as the general double auction mechanism presented in the previous section. Bids and asks are organized in lanes, and for each lane a the general double auction data structures are maintained.

Precisely, for each lane the MLDA keeps four structures to maintain bids sorted and another one global structure to buffer imprecise bids. In the literature many different data structures can be found for such an aim: Heaps, Internal Path Trees, AVL or sorted lists can be used, however, for the description of our algorithm we will consider the use of sorted lists due to they keep concepts simple and the used data structure 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 in the head of the list the highest losing bid. Sin is sorted in descending order, keeping in the head the highest winning ask and Sout is sorted in an ascending order to keep the lowest losing bid in its head.

By construction the MLDA keeps the Social Welfare maximum. This is the invariant of the insertion algorithms and is a key point for the economical efficiency of the MLDA.

The algorithm does not differentiate between precise or imprecise bids, for it 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 fundamental issue to be solved by our algorithms concerns to how impreciseness is handled. Note that if only precise bids 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 none bid in a lane can displace a bid in another lane. Contrarily, when there are imprecise bids 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 maximized.

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 maximum, so the choice of any of those situations is given by the condition that maximizes 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 update each time a bid or ask are inserted.

Using that pointers we can easily find the lane where the social welfare is maximized 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 maximize 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 PendingLosingBids queue. Contrarily, when the situation that maximizes the 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 aren't 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.

Ask insertion

We considered that asks cannot be imprecise because it does not make sense for a seller to offer imprecise time-specified resources. A seller will always indicate the specific resource that is selling including its specific time slot.

The ask insertion algorithm also has to maintain the invariant, that is, keep the social welfare maximum at each ask insertion. When an asks 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, 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 in order to be promoted as winning bid in L and matched with S. Due to our invariant, the selected bid has to be such bid that keeps the social welfare maximum. 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 algorithms selects the highest out of the possible. 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.

PendingLosingBids queue

The PendingLosingBids queue is a data structure that keeps bids organized by 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 at 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 arrives bids are removed from PendingLosingBids queue. Whenever no more asks arrive, bids are kept in the queue and considered to be losing bids.

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.

5.5 MLDA algorithm

5.5.1 Bid Insertion

```
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 ∅ ∧ LLAordered is ∅ ∧ HLBordered is ∅ then
        insertPendingSortedLosingBids (bnew)
        return
    if LLAordered is ∅ ∧ Bnew ≥ HLBmax then
        if Bnew > LWBmin then
            displaceLWB (Bnew, LWBmin)
        else
            insertPendingSortedLosingBids (Bnew)
    else if ¬ LLAordered is ∅ ∧ Bnew ≥ HLBmax then
        if LLAmin ≥ Bnew ≥ LWBmin then
            displaceLWB (Bnew, LWBmin)
        else if  LLAmin ≥ LWBmin ≥ Bnew then
            insertPendingSortedLosingBids (Bnew)
        else if LWBmin ≥ Bnew ≥ LLAmin then
            promoteLLA (Bnew, LLAmin)
        else if LWBmin ≥ LLAmin ≥ Bnew then
            insertPendingSortedLosingBids (Bnew)
        else if Bnew ≥ LLAmin ≥ LWBmin then
            displaceLWB (Bnew, LWBmin)
        else if Bnew ≥ LWBmin ≥ LLAmin then
            promoteLLA (Bnew, LLAmin)
        else
            return
    else if  ¬LLAordered is ∅ ∧ Bnew ≤ HLBmax then
        insertPendingSortedLosingBids (Bnew)
    else if LLAordered is ∅ ∧ Bnew ≤ HLBmax then
        insertPendingSortedLosingBids (Bnew)
    else
        return
end
```

Algorithm 1: Bid Insertion algorithm

```
Data: A Bid Bnew and Bid LLAmin
Result: LLAmin is promoted to Sin and matched with Bnew
begin
    k ← LLAmin.lane
    removeSout (LLAmin,k)
    insertBin (Bnew,k)
    insertSin (LLAmin,k)
end
```

Algorithm 2: PromoteLLA function.

```
Data: A Bid Bnew and Bid LWBmin
Result: LWB is displaced to pendingSortedLosingBids queu and Bnew inserted as
        a winner
begin
    k ← LWBmin.lane
    insertBin (Bnew,k)
    removeBin (LWBmin,k)
    insertPendingSortedLosingBids (LWBmin)
end
```

Algorithm 3: DisplaceLWB function.

5.5.2 Ask insertion

```
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 4: Ask insertion algorithm.

```
Data: Snew the ask to be inserted in L the target lane.
Result: Looks for a bid suitable to be paired with the ask to be inserted.
begin
    HWAOrdered ← List of the HWA ordered decreasing
    HWAOrderedSub ← is a sublist of HWAOrdered, such that all of its asks are bigger
    than Snew(Lj)
    HLBOrdered ← the list of highest loosing bids in all lanes and order it by decreasing
    values
    PendingBidsQueue ← Global queue containing all bids currently losing and pending to
    be inserted, sorted descending for each lane
    HLBmax ← HLBOrdered.head
    HWAmax ← HWAOrderedSub.head
    pendingMax ← PendingBidsQueue.head(L)
    if HLBmax is ∅∧ pendingMax is ∅ then
        if HWAOrderedSub is ∅ then
            insertSout (Snew,L)
        else
            changeHWAmax (Snew, HWAmax, L)
        end
        return
    end
    if HLBmax is ∅∨ HLBOrdered is ∅ then
        HLBmax ← pendingMax
    else if ¬ HLBmax is ∅∧ HLBmax < pendingMax then
        HLBmax ← pendingMax
    end
    if Snew ≤ HLBmax ∧ HWAOrderedSub is ∅ then
        if HLBmax is in a Lane then
            promoteHLB (Snew, HLBmax, L)
        else
            promoteHLBFromPending (Snew, HLBmax, L)
        end
    else if Snew ≥ HLBmax ∧¬ HWAOrderedSub is ∅ then
        if Snew > HWAmax then
            insertSout (Snew,L)
        else
            changeHWAmax (Snew, HWAmax, L)
        end
    else if Snew < HLBmax ∧¬ HWAOrderedSub is ∅ then
        if promoteOrDisplace (HWAmax, Snew, HLBmax) then
            if HLBmax is in a Lane then
                promoteHLB (Snew, HLBmax, L)
            else
                promoteHLBFromPending (Snew, HLBmax, L)
            end
        else if ¬ promoteOrDisplace (HWAmax, Snew, HLBmax) then
            if HLBmax is in a Lane then
                changeHWAmax (Snew, HLBmax, L)
            else
                promoteHLBFromPending (Snew, HLBmax, L)
            end

    else if Snew > HLBmax ∧¬ HWAOrderedSub is ∅ then
        insertSout (Snew,L)
    end
end
```

Algorithm 5: check bid promotion function

```
Data: an ask Snew an ask HWAmax and the Lane L
Result: HWAmax is removed from winning and the bid matched with it moved to
        L. Snew is also inserted as winning and HWAmax is re-inserted
begin
    t ← HWAmax.lane;
    HWB ← getHWBthatCanBeMovedToL (L,t);
    removeSin (HWAmax,t);
    removeBin (HWB,t); insertBin (HWB,L);
    insertSin (Snew,L);
    insertAsk (HWAmax);
end
```

Algorithm 6: changeHWAmax function

```
Data: Ask Snew, Bid Bmax, Lane L
Result: A bid Bmax is promoted from the PendingBidsQueue. Snew is inserted as
        winner in L.
begin
    if negHWA(L) is ∅∧ Bmax< HWA(L) vee Bmax < Snew then
        if HWA(L) > Snew then
            HWA ← HWA(L);
            insertSin (Snew,L);
            removeSin (HWA,t);
            insertSout (HWA,L);
        else
            insertSout (Snew,L);
        end
    else
        removeBid (PendingBidsQueue, bmax);
        insertBin (Bmax,L);
        insertSin (Snew,L);
    end
end
```

Algorithm 7: promoteHLBFromPending function

```
Data: Ask Snew, Bid HLBmax, Lane L
Result: A currently losing bid HLBmax is promoted from the losing set. Snew is
        inserted as winner in L. HLBmax is also inserted in L as winner.
begin
    if HLBmax<Snew then
        insertSout (Snew,L);
    else
        t ← HLBmax.lane;
        removeBout (HLBmax,t);
        insertBin (HLBmax,L);
        insertSin (Snew,L);
    end
end
```

Algorithm 8: promoteHLB function

5.5.3 Clearing

```
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 9: Clear algorithm.

5.6 Implementation and Experiments

MLDA has been implemented and a set of experiments have been carried out. The aim of the experiments 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

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

distribution functions have been used to generate random data. The distribution functions used were derived from several experiments found in the literature [64, 71, 72]. Uniform distribution of ask prices are motivated by the assumption that costs of resources are also uniformly distributed. Bid prices have been generated using different distribution functions, Normal distribution and Uniform distribution. Furthermore, bids and asks where also distributed across time slots following different distribution functions. Distributing bids and asks acroos different time slots using different distribution functions enabled us to experiment the effects of non-uniform supply/demand accross time slot. Every experiment describes the reason for the distribution function used and the aim of the intended evaluation.

Along the experiments we aimed to find the worst case of MLDA so as to determine the upper-bound of computation time. MLDA has been implemented as a set of sorted lists for each lane. Namely there are four sorted lists keeping respectively winning bids, losing bids, winning asks and losing asks. Besides, for each lane eight pointers are kept. The bid pointers point to the highest losing bid (HLB), highest winning bid(HWB), lowest winning bid (LWB) and lowest losing bid (LLB). The ask pointers keep the lowest losing ask (LLA), the highest losing ask (HLA), the lowest winning ask(LWA) and the highest winning ask (HWA). Bid insertion keeps three sorted lists of pointers (i.e. LWB, LWA, HLB) so as to facilitate the selection of the lane where the incoming bid should be inserted. Ask insertion keeps two lists of pointers (i.e. HWA and HLB). The algorithm also keeps a PendingLosingBids queue that buffers the bids that currently cannot be inserted but may be possibly inserted in a future. The buffer is structured as a set of lanes where for each candidate lane a bid can be inserted, a pointer in the corresponding lane of the PendingLosingBids queue is inserted. Displacing bids amongst lanes is a matter of moving pointers which is a non-costly operation. Lists are kept sorted, so the selection of candidate bids to be moved is also a constant cost operation. Due to the implementation of MLDA the most costly operation is to insert a bid into the PendingLosingBids queue since for each bid a pointer to the lanes where the bid is candidate to be placed is created and introduced in a sorted manner into the queue. Consequently our hypothesis is to consider that the worst case would be the one that maximizes the number of bids inserted in the PendingLosingBids queue. After some experimentation to validate our hypothesis, the worst case has been determined by the scenario characterized by the insertion of all bids first and subsequently all asks. By inserting first all bids without having introduced any ask, all bids are placed in the PendingLosingBids queue. As asks arrive bids are moved from the PendingLosingBids queue to their corresponding lanes. It can be thought that other bid placement strategies such as placing bids and asks alternatively and by increasing order of bid prices and decreasing order of ask prices will lead to worst computation times. However, this case makes that only the set of losing bids are inserted in the PendingLosingBids queue leading to a better computation time.

As a result of this hypothesis the experiments will be conducted using the worst case scenario for MLDA. That is, all bids will be inserted first and subsequently all asks will be introduced in the auction.

5.6.1 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 have been computed as:

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

Experiment A Setting

In order to evaluate the economical efficiency of the MLDA we aimed to compare with another well-known double auction, the k-Double Auction (k-DA). The k-DA implementation from the JASA framework [71, 72] have been taken to compare with our implementation of the MLDA. The JASA k-DA was based on the 4Heap Algorithm implementation presented by Bao et. al [11]. Experiments where conducted in a dual core T9500, 2.5GHz with 4Gb of Memory.

Experiment A description

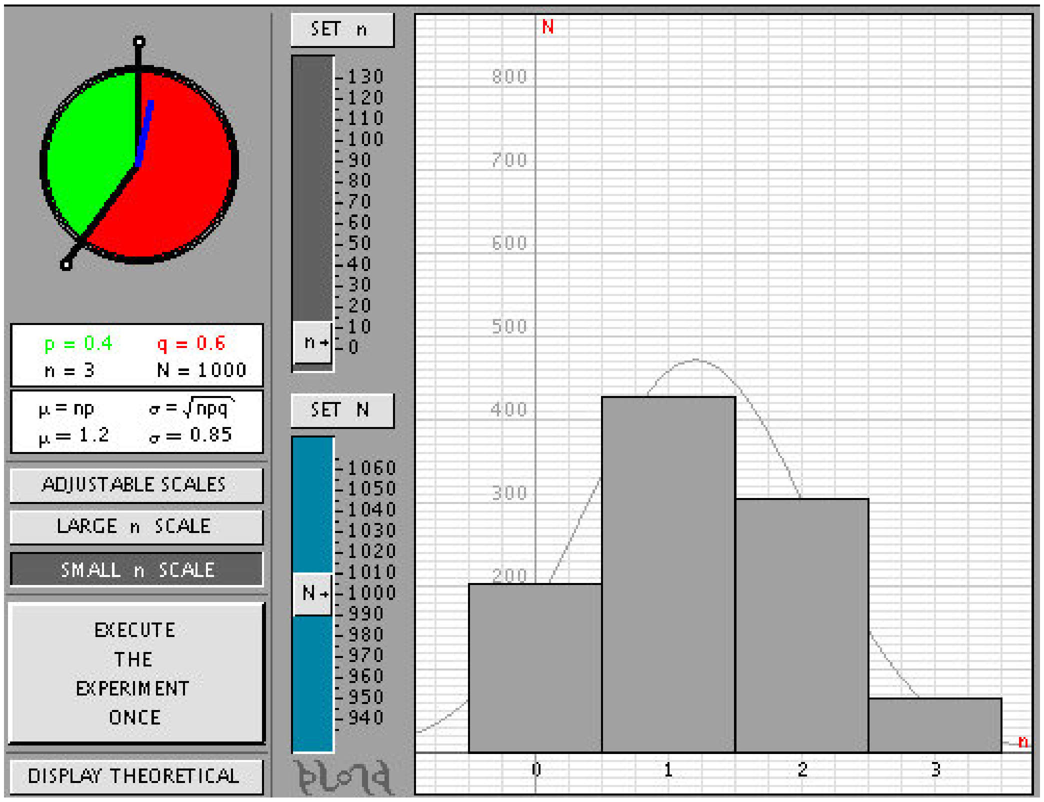
We carried 3 different experiments described in the following tables:

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)

Table 5.1: Experiment A.1 setting

Table 5.1 summarizes our first set of experiments with MLDA. The experiment consisted in the generation of a set of 1000 bids and 600 asks[2]. Several auction instances where created, one MLDA auction instance for four lanes, and three sets of four 4HeapDoubleAuction instances from JASA framework [71, 72]. Each set was referred as 4HDARR, 4HDAU, 4HDAN respectively.

[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.



p = 0.4 q = 0.6
n = 3 N = 1000
$\mu = np$ $\sigma = \sqrt{npq}$
$\mu = 1.2$ $\sigma = 0.85$

Figure 5.4: Binomial distribution of bids for 4HDAN.

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

For the case of 4HDARR, 4HDAU and 4HDAN, the same experiment was carried out. For 4HDARR, bids where inserted in a round robin fashion at 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... Asks were inserted following a uniform distribution amongst lanes. The experiment measured the time taken to compute the initialization of the four instances of the 4HeapAuction as well as the time taken to insert 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 (see Figure 5.4) respectively.

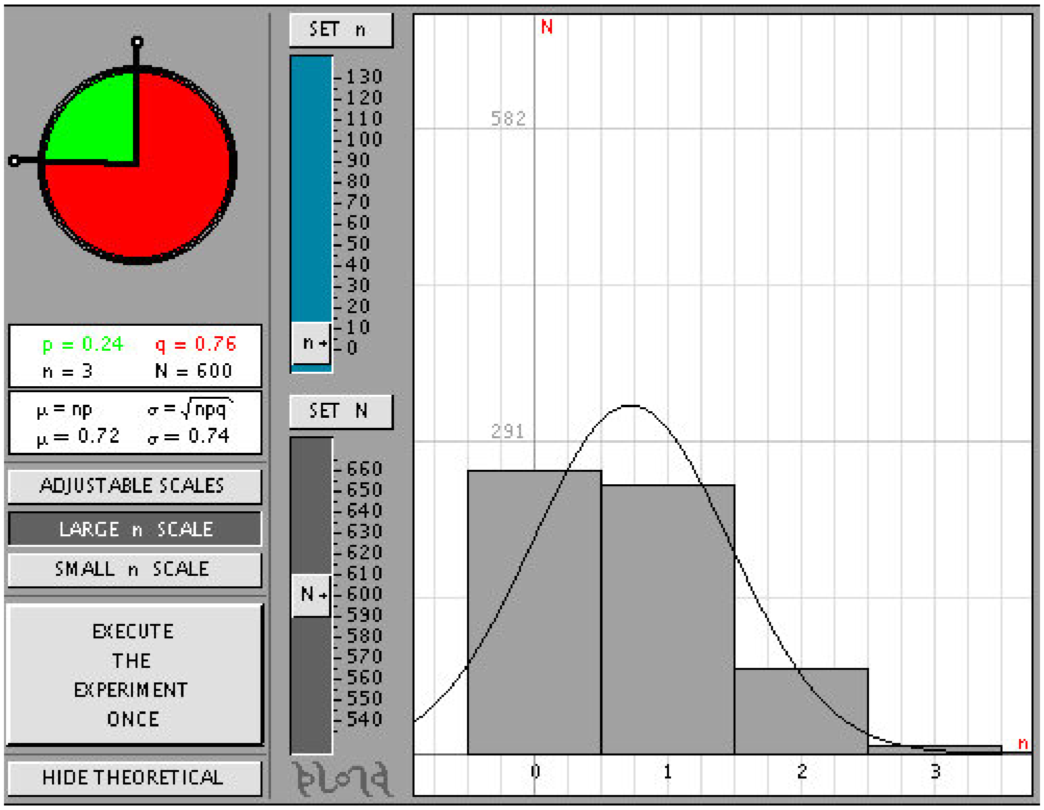
For the four experiments the same set of bids and asks where used in order to avoid divergences due to randomization. Experiments were repeated 100 times re-generating bids and asks at each experiment.

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)

Table 5.2: Experiment A.2 setting

Table 5.2 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 (see Figure 5.5). 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. Binomial distribution can be characterized as follow:

$$\Pr(K = k) = \binom{n}{k} p^k (1-p)^{n-k}$$
$$fork = 0, 1, 2, ..., n$$
$$s.t. \binom{n}{k} = \frac{n!}{k!(n-k)!}$$



p = 0.24 q = 0.76
n = 3 N = 600

$\mu = np$ $\sigma = \sqrt{npq}$
$\mu = 0.72$ $\sigma = 0.74$

Figure 5.5: Binomial distribution for asks.

Experiment A.3				
Attribute	**MLDA**	**4HDARR**	**4HDAU**	**4HDAN**
Repetitions	100	100	100	100
Lanes	4	4 instances	4 instances	4 instances
Bids	1000 (organized in subsets of lanes following a uniform distribution)	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)

Table 5.3: Experiment A.3 setting.

Table 5.3summarizes the setting for the third set of experiments. In A.3 we changed the placement of bids in MLDA. In previous tests we assumed that bids in MLDA where for all lanes (i.e., imprecise bids for all lanes), so a bid could be placed in any of the lanes of the auction. In this example we wanted to restrict this fact and placements of bids was generated following a uniform distribution amongst all possible combinations of lanes. For A.3 we generated 10 different groups of consecutive lanes, namely 1,2,3,4,12,23,34,123,234,1234 and bids where placed following a uniform distribution in one of this groups. The idea behind this test was to evaluate the efficiency obtained when some bids are restricted to some lanes and consequently MLDA is restricted when chosing the placement of bids.

5.6.2 Results Analysis

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 5.6 the average social welfare amongst the 500 experiments at each lane is the higher for MLDA. 4HDARR and 4HDAU achieve close to MLDA social welfare since bids and asks are distribute proportionally at each lane. However, the benefit of MLDA is that bids are always placed in the best option when 4HDARR and 4HDAU are restricted to place bids in one specific lane. 4HDAN achieves a worst social welfare due to unbalance of distribution between bids and asks.

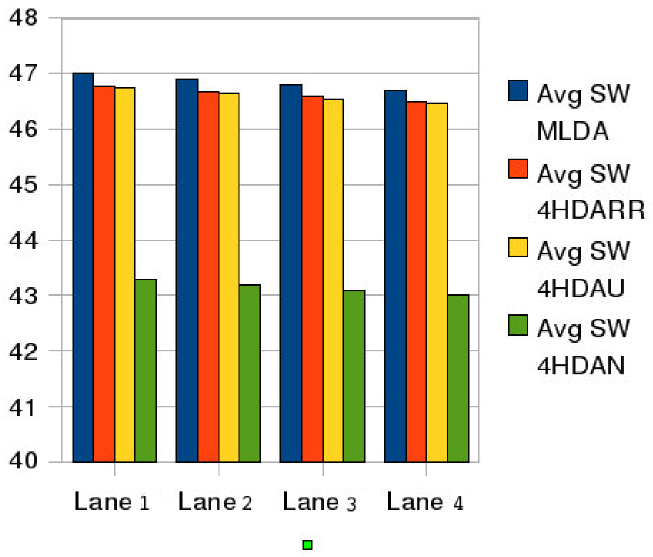
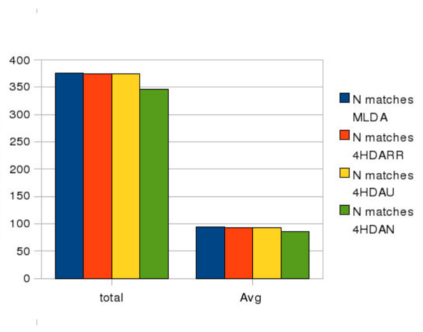


Figure 5.6: Compared average social welfare per lane. Y-axis indicates Social Welfare.

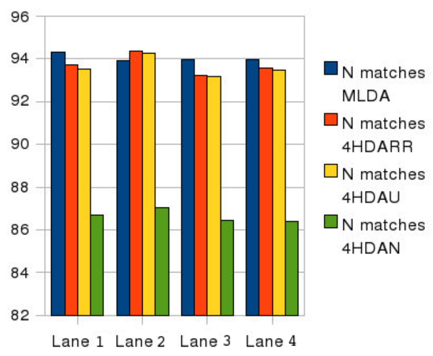
The same expected results when considering number of matches were encountered. As can be seen in Figure 5.7 MLDA achieved the higher number of matches, either per-lane and total. For the 4HDAN, we can see that the lane with highest probability achieves a

[3]see Section 5.4.4 for the description of the algorithm.

highest number of matches, but it still obtains the worst results.



(a) Matches per experiment



(b) Matches per lane

Figure 5.7: Number of matches per experiment and per lane. Y-axis indicates number of matches.

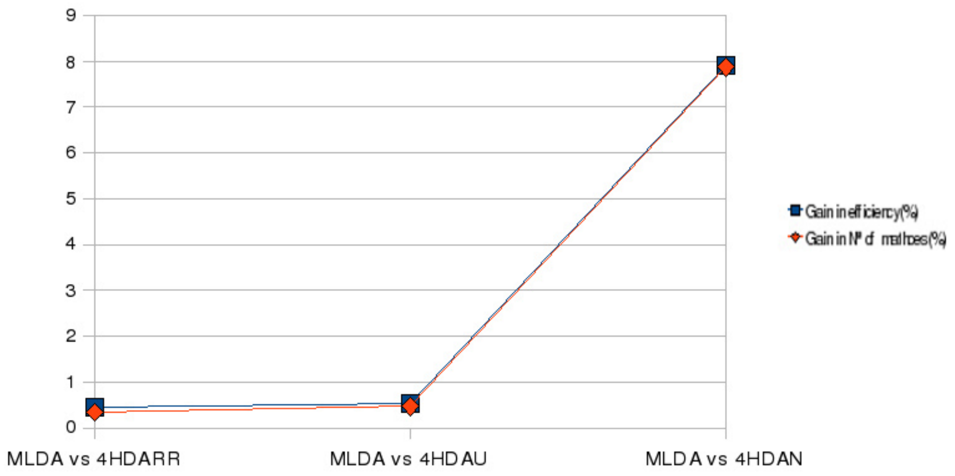


Figure 5.8: MLDA compared to 4HDA. Y-axis indicates improvement in %.

Figure 5.8 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 of the obtained social welfare and number of matches provided. MLDA is slighly a 0,5% better than 4HDARR and 4HDAU when asks are distributed following a uniform distribution amongst lanes. In this cases, the improvement obtained by MLDA is not very significant due to the distribution of bids in 4HDARR and 4HDAU which place bids almost uniformly across lanes. As asks and bids were placed following the same type of distribution, the number of matches per lane in 4HDARR and 4HDAU are close to the allocation obtained by MLDA. However, 4HDAN behaves poorly due to the distribution of bids being MLDA at least a 8% better that 4HDAN. This of course can be attributed to the unbalance between the distribution of asks and bids that let lanes 1 and 4 with a less number of bids which reduces the overall welfare and number of allocations. In this Figure it can also be seen a direct relation between the economic efficiency and the number of allocations. It can be deduced that the gain in efficiency is directly proportional to the gain in number of allocations.

5.6.3 Experiment A.2 Results

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

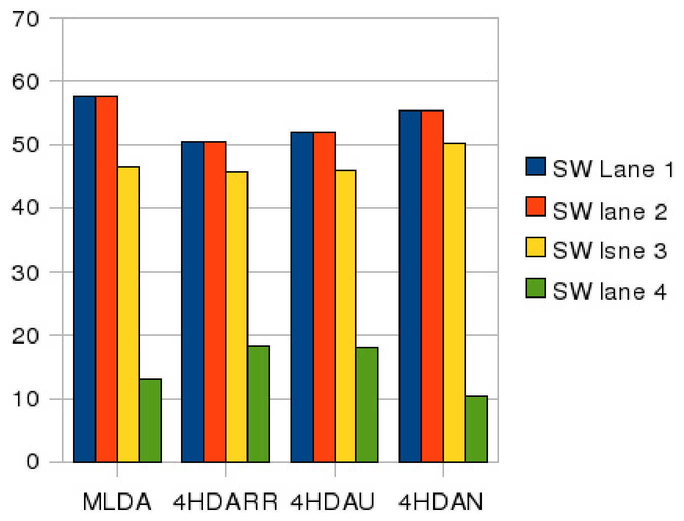


Figure 5.9: Compared average social welfare per lane. Y-axis indicates Social Welfare.

Figure 5.9 show that lanes 1 and 2 achieved higher social welfare compared to 3 and 4, this of course is due to the distribution of asks. Compared results show that MLDA gets the higher social welfare for lanes 1, 2 and 4 while 4HDAN achieves better social welfare at lane 3. Regarding average social welfare amongst lanes (see Figure 5.10) MLDA achieves the better social welfare followed by 4HDAN. 4HDAU and 4HDARR achieve worst social welfare due to the binomial distribution of asks and their almost equiproportional distribution of bids.

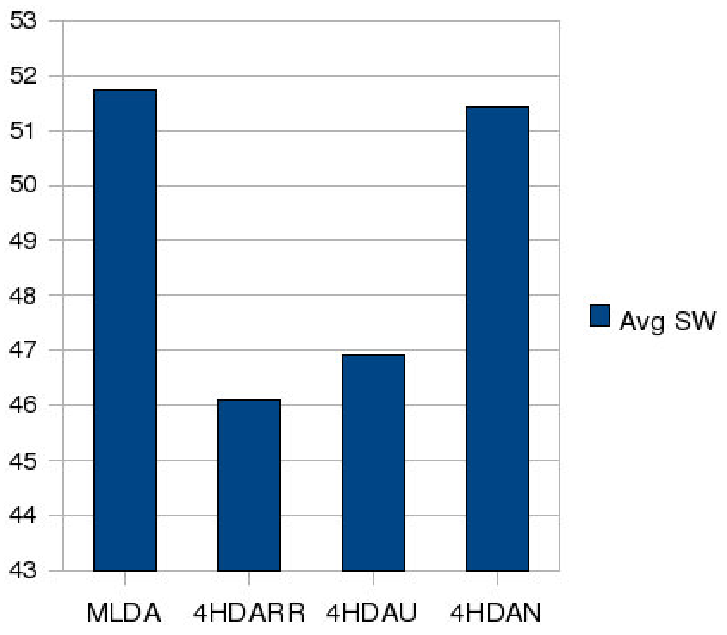


Figure 5.10: Average Social Welfare per experiment. Y-axis indicates Social Welfare.

Finally, Figure 5.11 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.

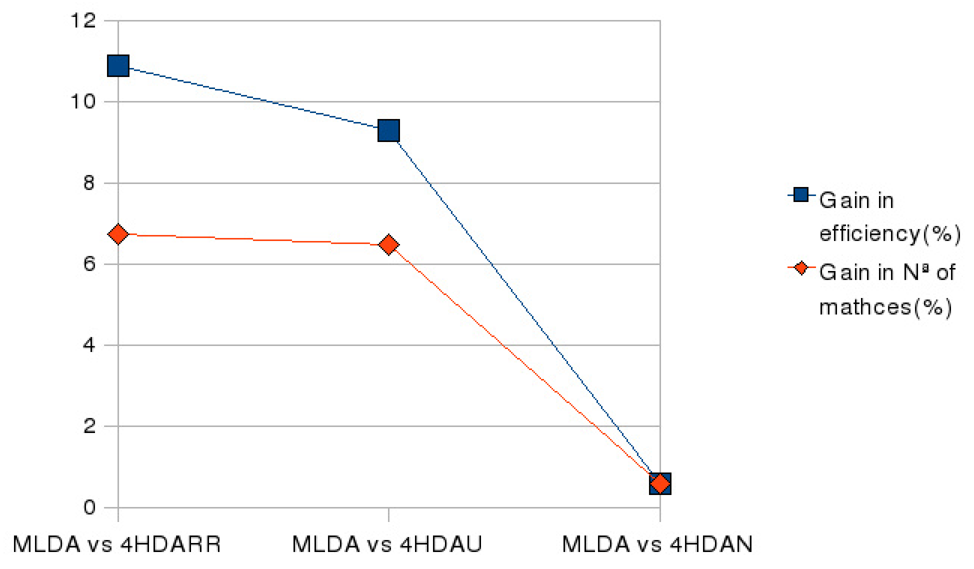


Figure 5.11: MLDA compared to 4HDA. Y-axis indicates improvement in %.

5.6.4 Experiment A.3 Results

As expected, MLDA achieved a better behaviour than other instances. Figure 5.12 shows the average social welfare and the average number of matches per lane obtained by the four auctions. It is clear that there are some bids in MLDA that were restricted to certain time slot (lane), however there are a certain quantity that has no restrictions and consequently can be placed in such a manner that social welfare is maximized. This flexibility is what makes MLDA obtain better results in terms of economical efficiency in respect to other auctions where bids are placed following a certain distribution. So, the worst case for MLDA would be the one where bids can be only placed in one lane. This would result in a behaviour very similar to the behaviour shown by 4HDARR,4HDAU,4HDAN depending on the distribution of bids inserted in MLDA. So we conclude that economically MLDA will behave in the worst case as 4HDA.

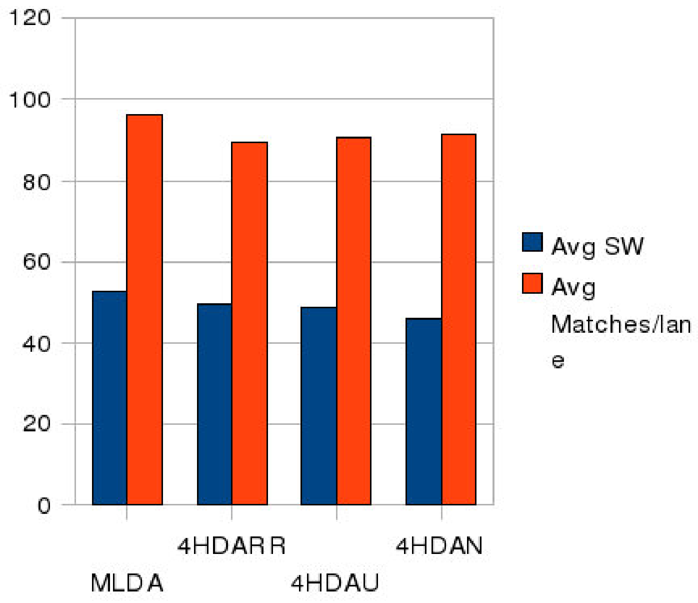


Figure 5.12: Average social welfare and matches per lane. Y-axis indicates Social Welfare.

Figure 5.13 presents the price per time slot. MLDA calculates a higher price per time slot than others due to a higher social welfare

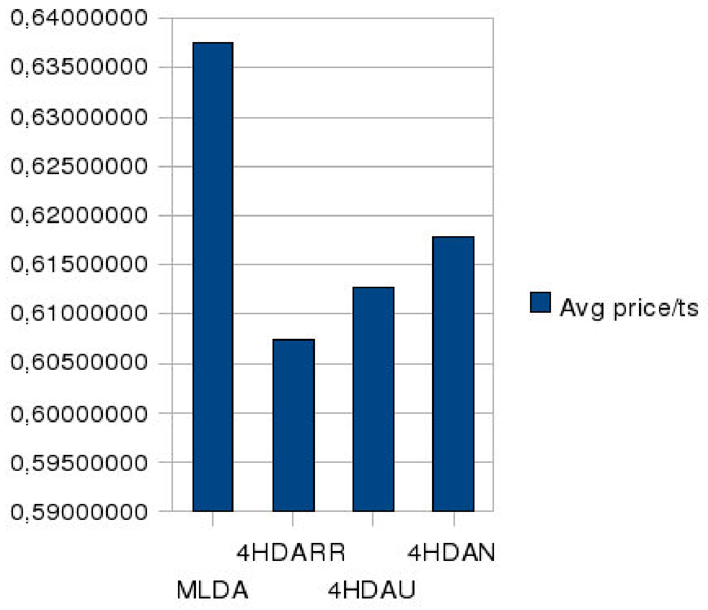


Figure 5.13: Average price per time slot. Y-axis indicates price.

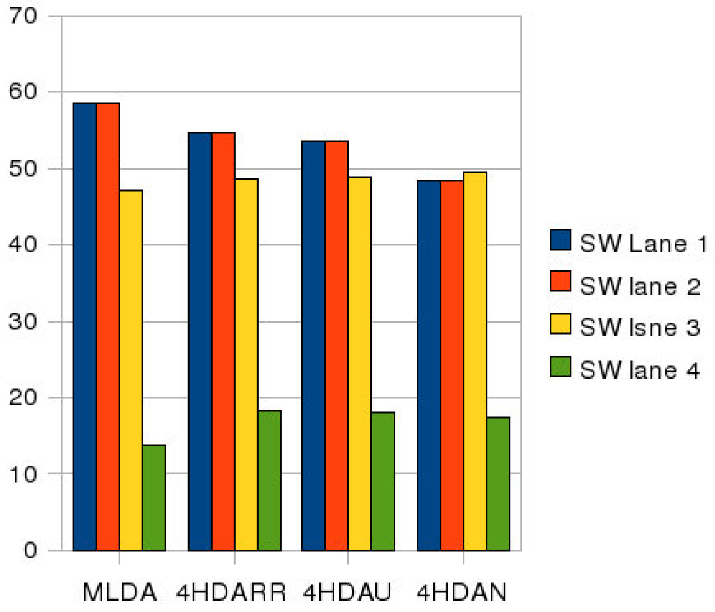


Figure 5.14: Social Welfare per lane. Y-axis indicates Social Welfare.

5.6.5 Experiment B: Computational Efficiency

Experiment B aimed to evaluate computational efficiency of MLDA. Experiments A.1, A.2 and A.3 have been used to obtain experimental data concerning the computational efficiency of the different auctions.

Time of computation in milliseconds have been used to determine the computation efficiency of the auction. For each experiment, the same operations were measured. MLDA was measured once data has been generated. Measurements started at MLDA instance creation and subsequent bid insertion. After insertion of asks and clearing the measurements ended. 4HDA (in any of their variants), where also measured after data generation. Namely, instance creation, bid and ask insertion and subsequent clearing where measured.

Results show that 4HDA settings take almost four times more time to finish the computation. We attribute the extra time to initialize different instances as well as its management. Besides we observe that when bids in MLDA are for a restricted set of lanes the time of computation is reduced due to a diminutions of the space of search of the MLDA algorithm (i.e. as more bids restricted to one lane, less number of bid searches in other lanes, besides, as less number of bids for all lanes, less times searches are for 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 5.15 shows the time taken to carry experiment A.1. It shows that MLDA is almost 4 times better than other 4HDA being the 4HDAN the worst case produced apparently by a higher number of bids in one instance in respect to others that produces a higher number of bid displacements.

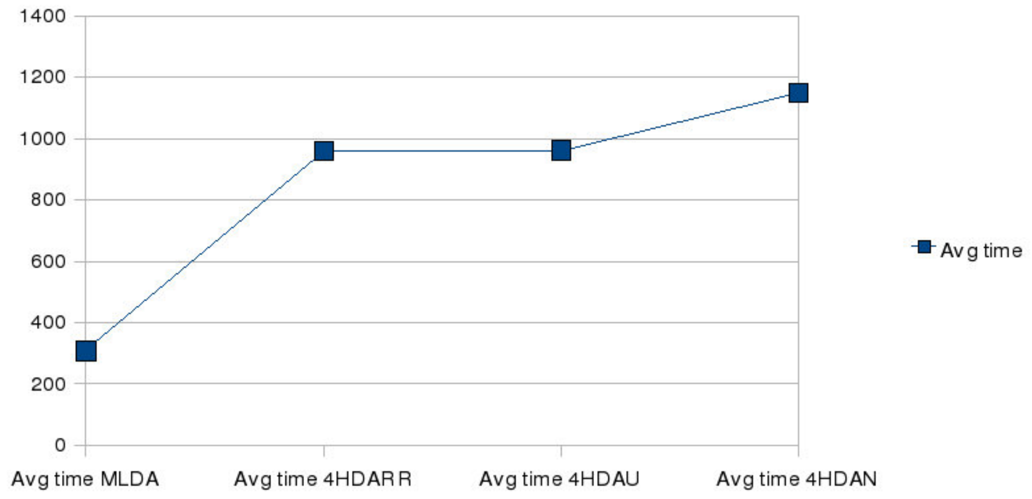


Figure 5.15: Time of computation for experiment A.1. Y-axis indicates time of computation in milliseconds.

Figure 5.16 shows similar results when asks are inserted following a binomial distribution.

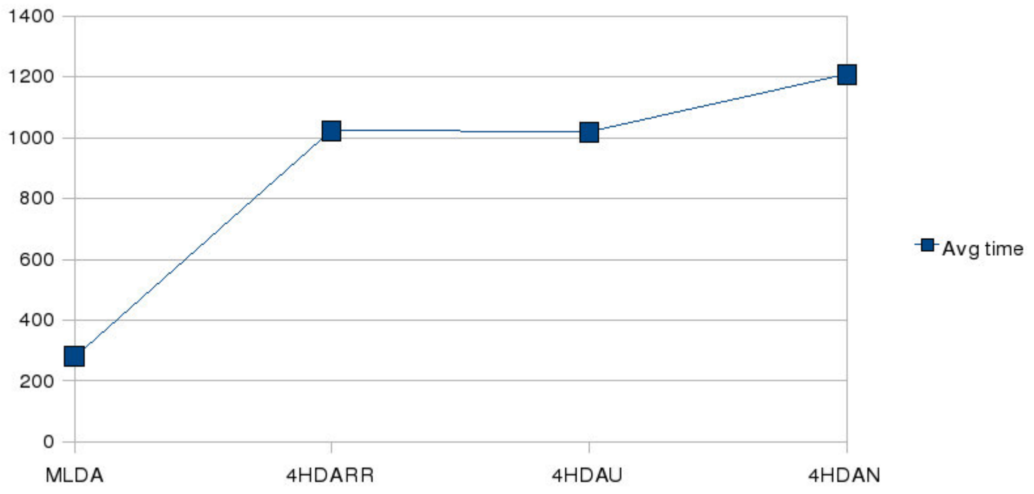


Figure 5.16: Time of computation for experiment A.2. Y-axis indicates time of computation in milliseconds.

Finally Figure 5.17 shows similar results for the three 4HDA auctions while MLDA behaves nearly ten times faster.

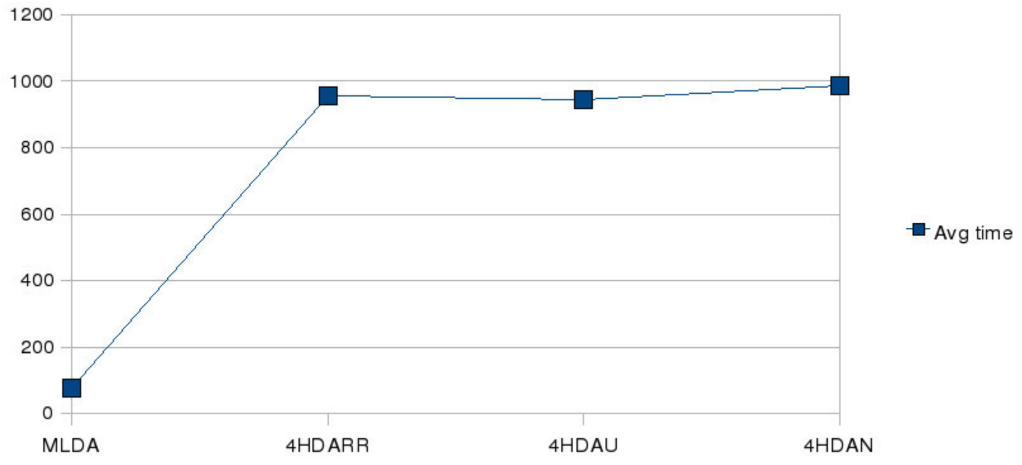


Figure 5.17: Time of computation for experiment A.3. Y-axis indicates time of computation in milliseconds.

5.6.6 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. Besides, we aimed to prove that the relation between the performance of MLDA and multiple instances of 4HDA are maintained.

The experiment has been defined as follow:

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)

Table 5.4: Experiment C.1 setting

The results obtained can be seen in Figure 5.18, it can be seen that our expectations were meet. MLDA keeps being 2 to 3 times faster as multiple auction instances of 4HDA. We measured the scalability of MLDA in comparison with others. One of the measures taken is the time increment between different number of lanes for each experiments. In average, MLDA increases the time of computation at around 250 ms being almost the same in other 4HDA experiments. We conclude that there is a linear increment of time as the number of lanes increases.

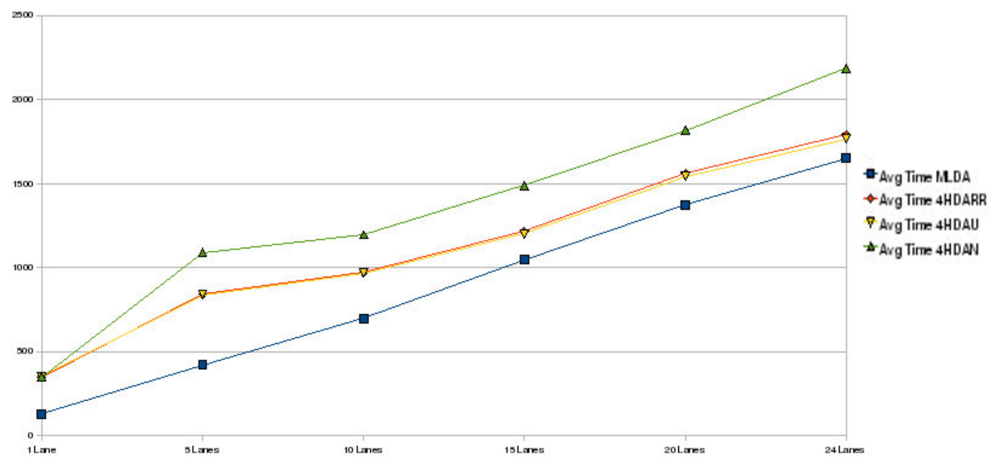


Figure 5.18: Compared average execution time for different number of time slots (lanes). Y-axis indicates time of execution in milliseconds.

5.6.7 Experiment D: Price per time slot

Until now, all experiments did not considered 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 per time slot. Previous experiments distributed bids following different distribution functions amongst 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 that costs can be assumed to be homogeneously distributed.

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(0.5,02)	1000 distributed round robin at lanes and prices distributed in a N(0.5,02)	1000 Uniformly distributed at lanes and prices distributed in a N(0.5,02)	10000 Following a Binomial PDF with n=3 and p=0,24 and prices distributed in a N(0.5,02)
Asks	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)	600 (Following a Uniform distribution)

Table 5.5: Experiment D.1 setting.

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. Besides, it is aimed to analyze the consequences of optimality[4] to the price per time slot. A priori it is not clear that the mechanisms that achieves best social welfare will achieve higher prices per time slot.

Besides, 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 resources to be allocated as soon as possible, so as near the time higher their willingness to pay for the resource. For that reason the experiment will place bids with higher prices close to the first offered time slot. Thus, the willingness of buyers to pay more to obtain earlier resources will be

[4] in social welfare terms

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:

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)

Table 5.6: Experiment D.2 setting.

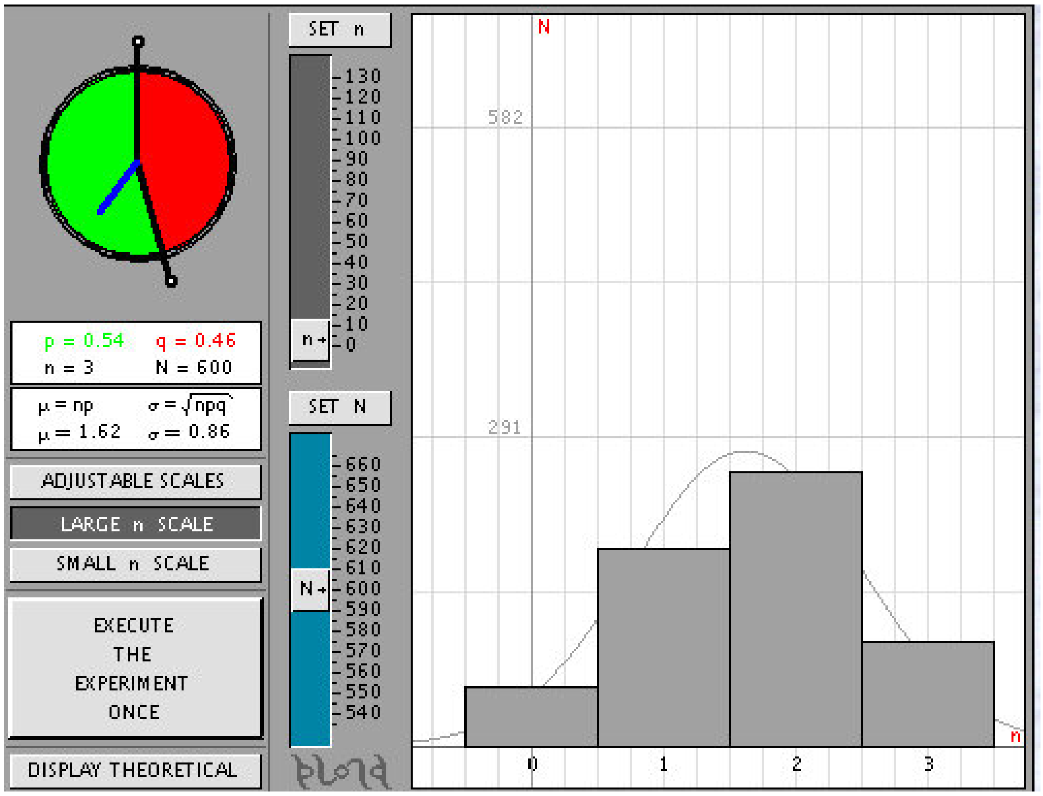


Figure 5.19: Binomial Distribution for asks in experiment D.2.

5.6.8 Experiment D.1 Results

As described in Table 5.5, the experiment set a Normal distribution of ask prices centered 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 5.20 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 maximized and because of the uniform distribution of asks that makes that the best configuration consists on distributing homogeneously bids 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. Worst results are obtained by 4HDAN due to the way bids are distributed. Lane 1 achieved a price per time slot very low 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 relation between the demand for a time slot and the final price achieved in that Lane when offer is fixed.

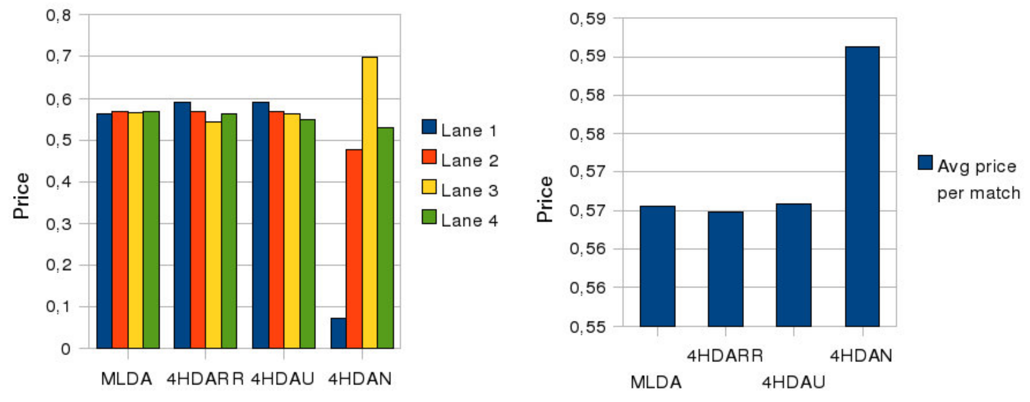


Figure 5.20: Distribution of prices per lane and prices per time slot. Y-axis indicates price.

On the right side of Figure 5.20 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 5.21 shows the number of matches per lane obtained by the fourth experiments. As already stated, MLDA achieves the higher number of matches while 4HDAN the worst due to the distribution of bids and asks.

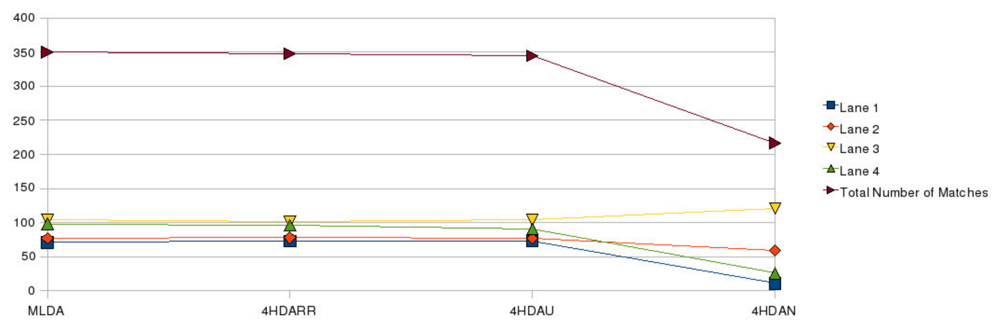


Figure 5.21: Matches per lane and total number of matches. Y-axis indicates number of matches.

5.6.9 Experiment D.2 Results

As described in Table 5.6 the experiment aimed to price more the first two time slots (lanes) to simulate the willingness of buyers to allocate resources 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.

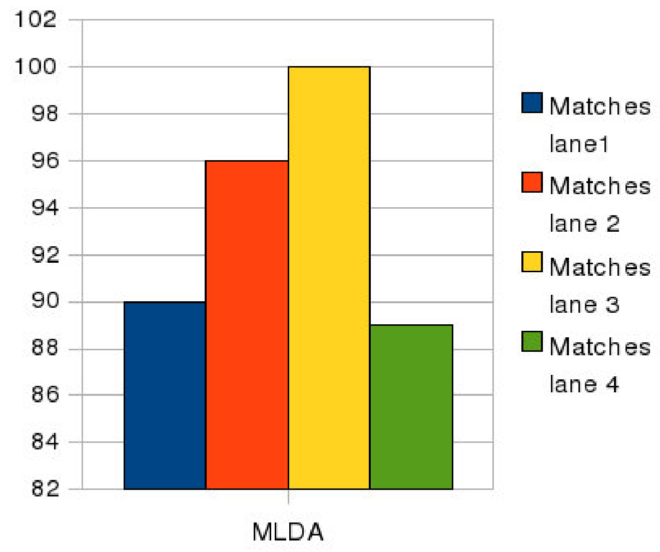


Figure 5.22: Matches per lane. Y-axis indicates number of matches.

Figure 5.22 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 asks. Lane two obtains a less amount of matches even prices are higher due to a less quantity of asks in that lane.

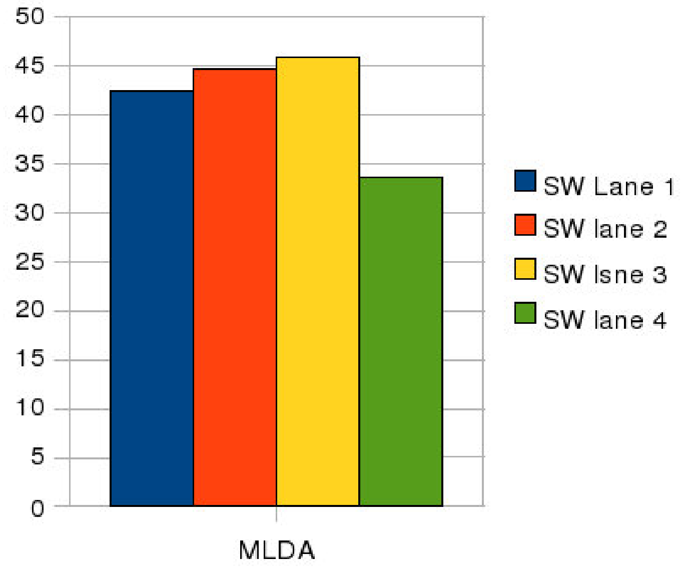


Figure 5.23: Social Welfare per lane. Y-axis indicates social welfare.

Figure 5.23 shows the distribution of the social welfare generated by MLDA amongst lanes. Asks distribution guide the number of matches per lane being 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 5.24. Lane 2 even having a higher number of matches, which would mean less price per time slot, achieves a similar price per time slot as lane one that achieves the maximum due to a short offer and higher bid prices.

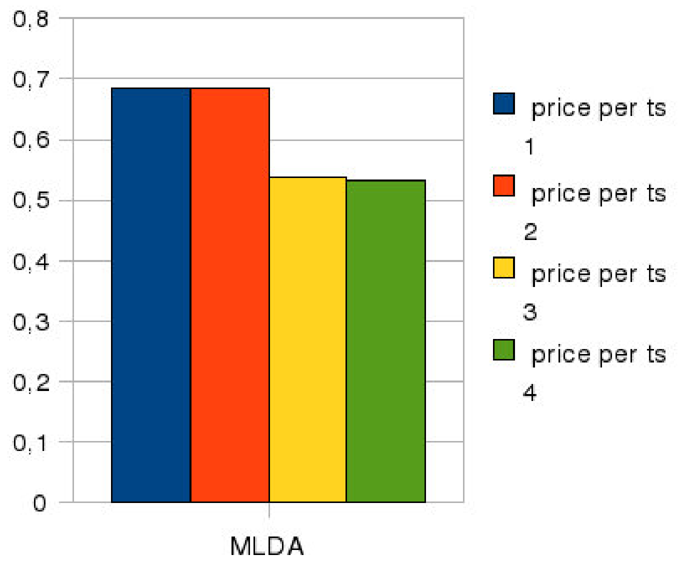


Figure 5.24: Price per time slot. Y-axis indicates price.

It can be concluded that ask distribution constraints 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 as higher the bid price, higher the number of matches when asks have uniform prices.

5.7 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 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 the data structures used in the implementation. For this experiments is not important the total amount of memory used by both instances but the relation between the amount of memory used. What it 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 5.7 summarizes the experiment setting that consisted in 500 iterations of an experiment that inserted 1000 bids and 600 asks to an instance of every one of the evaluated auctions.

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)

Table 5.7: Experiment E.1 setting.

Memory usage has been measured during the insertion of 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 garbage collector have 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.

5.7.1 Experiment E Results

Results of the 500 experiments can be seen in Figure 5.25. 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. This 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 5.26 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 for each bid a pointer to the lanes where a bid can be placed, and to the sorted lists used to maintain the different quotes across lanes.

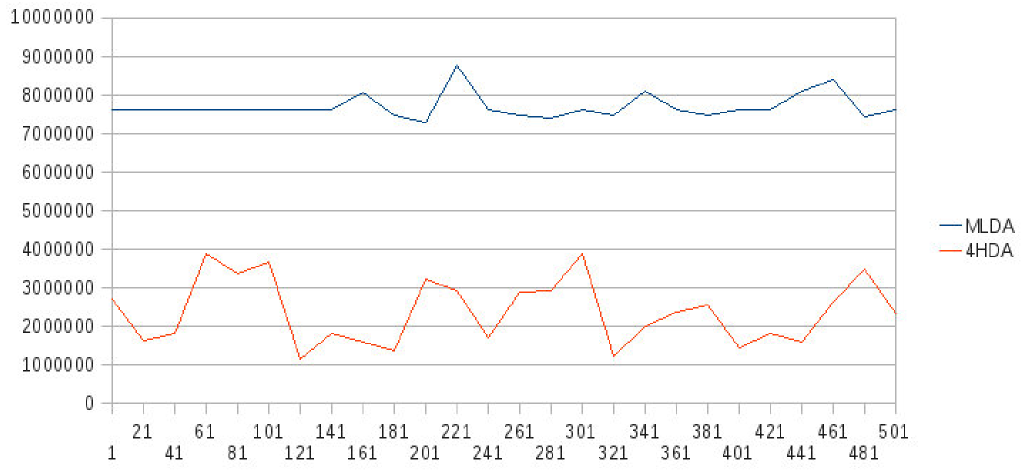


Figure 5.25: Memory usage during 500 experiments. Y-axis indicates the amount of memory used in bytes.

MLDA achieves better computational performances as demonstrated in previous experiments at expenses 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.

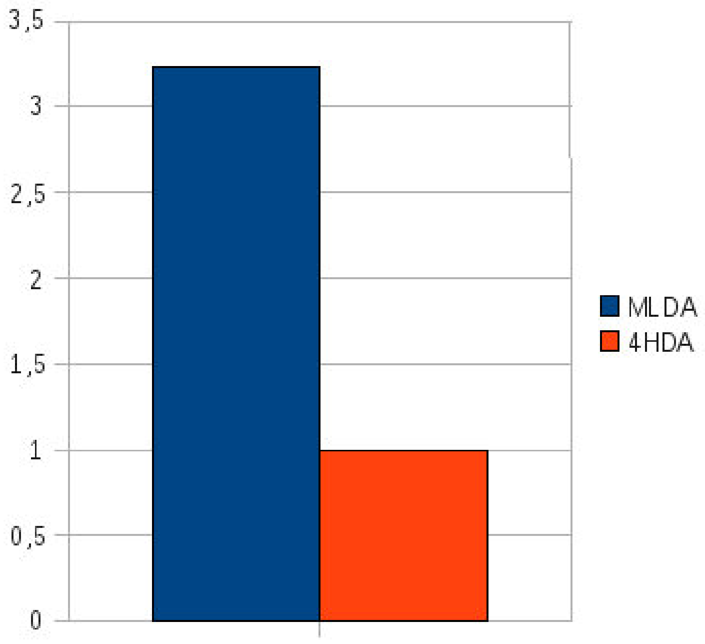


Figure 5.26: Compared memory usage between MLDA and 4HDA. Y-axis indicates the relation of the amount of memory required.

5.8 Conclusions

The chapter presented MLDA a novel variant of the well known Double Auction adapted to trade time-differentiated resources. The chapter motivated the utility of the auction as well as it set the context to be applied. MLDA is the final step in the research carried out in this thesis and constitutes an important contribution to build scalable and Grid oriented resource marketplaces. The chapter 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 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. Results shown a better behaviour of MLDA in all situations due to the properties of the auction (i.e. invariant maximum social welfare). Besides, MLDA has been compared with several double auction instances running at a 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. As a drawback, MLDA requires three times more memory than the other double auctions used in the evaluation due to the extra data structures used by MLDA to buffer not matched bids and the sorted lists to keep quotes per lane. Finally, MLDA showed to be a good candidate auction to trade time-differentiated resources 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 computational costs of CAs. Next step on evaluation will compare MLDA with an instance of a Combinatorial Auction.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Utility computing has shown to be a promising field of investment where large companies are developing their business models. Until now, distributed-computation and massive storage have been mainly used by reduced communities such as scientists, chemical companies, IT companies and research centres and universities. This thesis envisaged the gathering and provision of distributed-computing capacities to Internet users. Adhoc communities and small organizations are being created through the use of emerging paradigms such as the peer-to-peer. Massive computation can be attained by the aggregation of computational capacities from the extremes of Internet, taking advantage of idle resources and increasing broadband access. The thesis dealt with some of the main issues related to resource allocation in environments characterized by small communities that share the resources of their participants and require mechanism to increase their capacities under demand. This scenarios where characterized and referred as Contributory Systems. While internal Virtual Organization (VO) resources, mainly contributed by their participants, are managed by VO policies and scheduling services, external resources are traded in markets. The thesis aimed to study three main aspects related to the on-demand expansion of Virtual Organization resources. First, the description and specification of the traded resources using a generic and flexible language that can support evolution and heterogeneity. Second a peer-to-peer based architecture to manage, expand and share resources belonging to a VO. And third, an auction specially designed to trade time-differentiated resources as computational resources are. The economical approach taken in this thesis is motivated by the fact that resource allocation, when resources are scarce, need to be optimal in the sense that resource should be given to these that need them more. Many different approaches can be taken, but economics have shown that price can be used as a metric to express the QoS requirements, and consequently the utility obtained by a resource can be expressed through a price. Price will be used to determine to who resources have to be allocated obtaining then optimal behaviour.

The thesis dealt with semantics of Grid resources identifying their generic properties. A semantic approach aimed to characterize resources so as to build a common knowledge to make resource description independent of specific resources and technological evolution. Besides, a flexible and generic bidding specification have been proposed. The specification aimed to be a tool to express bidders preferences indifferently of the type of markets and scenario. It has been outlined the need of market co-habitation as means to generic resource allocation in scenarios characterized by diversity, heterogeneity, evolution and dynamism. The proposed specification has been used in the Grid4All European project as the bidding language used by buyer and seller agents. Besides, the resource description have been used to develop the Grid4All ontology, used by the Semantical Information Services that enables discovery of markets.

Next, the thesis proposed an architectural approach to decentralized resource allocation in Contributory Systems. DyMRA, a peer-to-peer based architecture has been proposed as a brokering infrastructure to negotiate for external resources. DyMRA was designed as a set of autonomous components and protocols to provide the functionality of on-demand expansion of resources in Virtual Organizations. DyMRA proposed the use of market mechanisms to mediate the allocation of resources. DyMRA has not been bound to an specific type of market but contrarily it relied in standard APIs and functionalities so as to support multiple co-habitating mechanism. DyMRA is currently implemented as an extension of LaCOLLA middleware available at http:// www.lacolla.uoc.edu/lacolla.

Markets where held by CAS, a configurable auction server architecture specially designed to support multiple market mechanisms. CAS has been designed as a set of configurable components following the separation of concerns paradigm. In CAS generic functionalities have been separated from specific market implementations providing a set of configurable components and binding functionalities that permit on-demand and dynamic market configuration and deployment. CAS offered well defined APIs to enable market mechanism developers to easily implement specific market mechanisms. DyMRA and CAS where bound together and constitute a totally decentralized approach to external resource provision and expansion in Virtual Organization. Besides CAS has been integrated in the Grid Marketplace (GRIMP) component of the Grid4All project and we are currently working in its evaluation.

Finally, an specific market mechanism have been designed. The approach has been motivated because the unsuitability of current auctions to allocate efficiently time-differentiated resources (usually provided by many different resource providers) such as most of the resources in a Grid. Even Combinatorial Auctions enable the trade of time-differentiated resources, they have a high computational cost. Simpler auctions do not enable multi-item trading but only one type of item is allocated. Straightforward approaches require multiple instances of single item auctions to manage time-differentiated resources. To cope with that issue the Multi-Lane Double Auction was proposed (MLDA). A MLDA is a double auction specially adapted to trade time-differentiated resources. Time-differentiated resources have been organized in lanes and bidders submit their bids for substitute time-slots letting the auction to calculate optimal allocations. MLDA has been compared with multiple instances of single item double auctions and several improvements were shown. On the one hand, MLDA achieved better economical efficiency than double auction in the presence of substitute or not precise bids. By design MLDA placed bids in the lane where the social welfare was maximized. On the other hand, in terms of computational efficiency, the tests carried out showed that MLDA ran two to three times faster than multiple instances of single item double auctions. That improvement was attributed to the costs of maintaining different instances running instead of a single one used by MLDA. As CAS, MLDA is one of the mechanisms implemented in the Grid4All project.

A concluding remark of the overall work carried out in this thesis is that of requirement for open APIs and layered software architectures. Dynamics and scale are important issues to be tackled by decentralization and flexibility. In the overall work of the thesis a set of requirements have always been considered, Ad-Hoc Grids and Contributory Systems are dynamic environments by nature, where multiple types of resources, services are exposed. Users and applications have varying resource needs and strategies. To provide functionalities to mediate the allocation of resources, we considered that specific implementations were not useful but instead generic components were required. Generic components should provide tools and functionalities to be adapted and configured to the addressed scenarios. To clearly understand the scenario and its requirements a clear comprehension of the properties of the resources is needed as well. The objective of this work had been to stablish the basis for long term architectures that can be implanted in the next generation of Internet.

6.2 Future Work

Future directions include work on strategy of auctions. Application requirements have to be translated to jobs and subsequently to a bid formulation. Bid formulation have to take into account the real available resources. Application requirements can be mapped to different configurations and bidding agents need to find configurations that maximize user's utility within certain budget constraints. Bidding strategies are also fundamental for a correct behaviour of markets. Bidding agents need to act based on their own strategy, that is, placing bids at certain times, deciding the price to bid based on the utility they expect and selecting the more appropriate type of auction to bid in.

Other interesting aspects that will be studied are the construction of an overall economic architecture, integrating some of the existing work on decentralized market information services [13], accounting and banking services [58] and reputation schemes [50] to provide a fully decentralized economic framework for resource allocation in Ad-hoc Grids.

Other open issues related to semantics are the description of markets and their properties and the provision of languages to describe running markets and their protocols. On the one hand, market semantics can be used to build a component repository of market implementations and a language to retrieve and instantiate specific market mechanisms. This will enhance the flexibility of resource allocation frameworks providing tools to initiate any market implementation at will. For example, eBay [31] and other trading sites can also be interested in such type of repositories enabling market initiators to select amongst a large set of auction implementations. Besides, market developers can upgrade the functionalities of trading sites by providing new market implementations.

Bibliography

[1] A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32(2):135–164, 2002.

[2] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, September 2000.

[3] Jeannie Albrecht, David Patterson, and Amin Vahdat. Distributed resource discovery on planetlab with sword. In *In WORLDS - First Workshop on Real, Large Distributed Systems*, 2004.

[4] Bruce Allen. Einstein@home. http://einstein.phys.uwm.edu/.

[5] Nejla Amara-Hachmi, Xavier Vilajosana, Ruby Krishnaswamy, Leandro Navarro-Moldes, and Joan Manuel Marquès. Towards an open grid marketplace framework for resources trade. In *OTM Conferences (2)*, pages 1322–1330, 2007.

[6] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE computer society, 2004.

[7] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. Ourgrid: An approach to easily assemble grids with equitable resource sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, Seattle, WA, USA, June 2003.

[8] Yedidia Atzmony and David Peleg. Distributed algorithms for english auctions. In *DISC '00: Proceedings of the 14th International Conference on Distributed Computing*, pages 74–88, London, UK, 2000. Springer-Verlag.

[9] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures, 2004.

[10] Shengli Bao and Peter R. Wurman. A comparison of two algorithms for multi-unit k-double auctions. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 47–52, New York, NY, USA, 2003. ACM Press.

[11] Shengli Bao and Peter R. Wurman. A comparison of two algorithms for multi-unit k-double auctions. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 47–52, New York, NY, USA, 2003. ACM.

[12] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. *Softw. Pract. Exper.*, 36(11-12):1257–1284, 2006.

[13] René Brunner, Isaac Chao, Pablo Chacin, Felix Freitag, Leandro Navarro, Oscar Ardaiz, Liviu Joita, and Omer F. Rana. Assessing a distributed market infrastructure for economics-based service selection. In *OTM Conferences (2)*, pages 1403–1416, 2007.

[14] Ali Raza Butt, Rongmei Zhang, and Y. Charlie Hu. A self-organizing flock of condors. In *SC*, page 42. ACM, 2003.

[15] R. Buyya and S. Venugopal. The gridbus toolkit for service oriented grid and utility computing: An overview and status report, 2004.

[16] Rajkumar Buyya, David Abramson, and Jonathan Giddy. An economy driven resource management architecture for global computational power grids. In Hamid R. Arabnia, editor, *PDPTA*. CSREA Press, 2000.

[17] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks. In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC*, pages 140–145, New York, NY, USA, 2002. ACM Press.

[18] 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.

[19] Ruggiero Cavallo, David C. Parkes, Adam I. Juda, Adam Kirsch, Alex Kulesza, Sébastien Lahaie, Benjamin Lubin, Loizos Michael, and Jeffrey Shneidman. Tbbl: A tree-based bidding language for iterative combinatorial exchanges. In *Multidisciplinary Workshop on Advances in Preference Handling (IJCAI)*, 2005.

[20] Jeffrey S. Chase, David E. Irwin, Laura E. Grit, Justin D. Moore, and Sara E. Sprenkle. Dynamic virtual clusters in a grid site manager. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 90, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational resource exchanges for distributed resource allocation.

[22] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational resource exchanges for distributed resource allocation. Technical report, 2004.

[23] Scott H. Clearwater, editor. *Market-based control: a paradigm for distributed resource allocation*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[24] research & Development Information Service Community. The Future Network and Services Draft. Technical report, An information space for European Research and Development (R&D) and exploitation of European R&D results, 2007.

[25] Grid4All Consortium. Grid4all european project. http://grid4all.eu/.

[26] Sven de Vries and Rakesh V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

[27] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 4(5):63–74, 2000.

[28] Zoran Despotovic, Jean-Claude Usunier, and Karl Aberer. Towards peer-to-peer double auctioning. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, page 90289.1, Washington, DC, USA, 2004. IEEE Computer Society.

[29] distributed.net. urlhttp://www.distributed.net/.

[30] Scott Draves. The Electric Sheep and their Dreams in High Fidelity. pages 7–9, New York, 2006. ACM Press.

[31] eBay. ebay web site. http://www.ebay.com/.

[32] Marc Esteva and Julian A. Padget. Auctions without auctioneers: Distributed auction protocols. In *Agent Mediated Electronic Commerce (IJCAI Workshop)*, pages 220–238, 1999.

[33] T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, and F. Freitag. Decentralized resource allocation in application layer networks. In *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid (CCGrid 2003*, pages 645–650, 2003.

[34] Michal Feldman, Kevin Lai, and Li Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. *CoRR*, abs/cs/0502019, 2005.

[35] Folding. Folding@home distributed computing. http://folding.stanford.edu/.

[36] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[37] Daniel Friedman. The double auction market institution: A survey. In Daniel Friedman and John Rust, editors, *The Double Auction Market*, pages 3–25. Addison-Wesley, 1993.

[38] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: An architecture for secure resource peering. In *In Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 133–148, 2003.

[39] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553. Morgan Kaufmann Publishers Inc., 1999.

[40] Ali Ghodsi. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH—Royal Institute of Technology, Stockholm, Sweden, October 2006.

[41] Ali Ghodsi. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH—Royal Institute of Technology, Stockholm, Sweden, December 2006.

[42] Thomas R. Gruber. A translation approach to portable ontology specifications, 1993.

[43] D. Haussheer and B. Stiller. Decentralized auction-based pricing with peermart. In *Integrated Network Management*, pages 381–394. IEEE, 2005.

[44] D. Haussheer and B. Stiller. Decentralized auction-based pricing with peermart. In *Integrated Network Management*, pages 381–394. IEEE, 2005.

[45] Holger H. Hoos and Craig Boutilier. Solving combinatorial auctions using stochastic local search. In *AAAI/IAAI*, pages 22–29, 2000.

[46] http://lacolla.uoc.edu/lacolla/.

[47] http://setiathome.berkeley.edu/.

[48] http://www.mersenne.org/prime.htm.

[49] David Irwin, Jeffrey Chase, Laura Grit, Aydan Yumerefendi, David Becker, and Kenneth G. Yocum. Sharing networked resources with brokered leases. In *USENIX Annual Technical Conference*, pages 199–212.

[50] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-molina. The eigentrust algorithm for reputation management in p2p networks. In *In Proceedings of the Twelfth International World Wide Web Conference*, pages 640–651. ACM Press, 2003.

[51] F. Kelly. Charging and rate control for elastic traffic, 1997.

[52] Bernardo A. Huberman Kevin Lai and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical Report arXiv:cs.DC/0404013, HP Labs, Palo Alto, CA, USA, April 2004.

[53] Konstantinos Kotis and A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006.

[54] Vijay Krishna. *Auction Theory.* Academic Press, 2002.

[55] Anthony M. Kwasnica, John O. Ledyard, Dave Porter, and Christine DeMartini. A new and improved design for multiobject iterative auctions. *Manage. Sci.*, 51(3):419–434, 2005.

[56] Daniel Lázaro, Joan Manuel Marquès, and Josep 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.

[57] Daniel Lázaro, Xavier Vilajosana, and Joan Manuel Marquès. Dymra: Dynamic market deployment for decentralized resource allocation. In *OTM Workshops (1)*, pages 53–63, 2007.

[58] Xavier León. Currency Management System: a Distributed Banking Service for the Grid. Technical Report UPC-DAC-RR-XCSD-2007-6, Universitat Polit
'ecnica de Catalunya, Barcelona, Spain, July 2007.

[59] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[60] Joan Manuel Marquès, Xavier Vilajosana, Thanasis Daradoumis, and Leandro Navarro. Lacolla: Middleware for self-sufficient online collaboration. *IEEE Internet Computing*, 11(2):56–64, 2007.

[61] Joan Manuel Marquès, Xavier Vilajosana, Thanasis Daradoumis, and Leandro Navarro. Lacolla: Middleware for self-sufficient online collaboration. *IEEE Internet Computing*, 11(2):56–64, 2007.

[62] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.

[63] R. Preston McAfee and John McMillan. Auctions with entry. *Economics Letters*, 23(4):343–347, 1987.

[64] Kevin L. Mills and Christopher Dabrowski. Can economics-based resource allocation prove effective in a computation marketplace? *Journal of Grid Computing*, 6:291–311, Sept 2008.

[65] Muralidhar V. Narumanchi and José M. Vidal. Algorithms for distributed winner determination in combinatorial auctions. In *Agent-Mediated Electronic Commerce VII*, 2005.

[66] Dirk Neumann, Jochen Stößer, Arun Anandasivam, and Nikolay Borissov. Sorma - building an open grid market for grid resource allocation. In Jörn Altmann and Daniel Veit, editors, *GECON*, volume 4685 of *Lecture Notes in Computer Science*, pages 194–200. Springer, 2007.

[67] Noam Nisan. Bidding and allocation in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[68] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. 613.

[69] P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. Frank, and C. Chokkareddy. Ocean: The open computation exchange and arbitration network, a market approach to meta computing, 2003.

[70] David W. Pearce. *The MIT dictionary of modern economics.* MIT Press, 1986.

[71] Steve Phelps. Web site for JASA (Java Auction Simulator API), 2006. `http://www.csc.liv.ac.uk/~sphelps/jasa/` (accessed February 23, 2006).

[72] Steve Phelps. *Evolutionary Mechanism Design.* Ph. D thesis, University of Liverpool (U.K.), 2007.

[73] Martin Placek and Rajkumar Buyya. Storage exchange: A global trading platform for storage services. In *Euro-Par*, pages 425–436, 2006.

[74] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *HPDC '98: Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, page 140, Washington, DC, USA, 1998. IEEE Computer Society.

[75] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[76] Ori Regev and Noam Nisan. The popcorn market—an online market for computational resources. In *ICE '98: Proceedings of the first international conference on Information and computation economies*, pages 148–157, New York, NY, USA, 1998. ACM.

[77] Donald Rutherford. *Dictionary of economics.* Routledge, 1992.

[78] Tuomas Sandholm. An algorithm for winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, February 2002.

[79] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *AAMAS*, pages 69–76. ACM, 2002.

[80] Luis F. G. Sarmenta. Bayanihan: Web-based volunteer computing using java. In *In Second International Conference on World-Wide Computing and its Applications*, pages 444–461, 1998.

[81] Björn Schnizler. Mace: A multi-attribute combinatorial exchange. In Nick Jennings, Gregory Kersten, Axel Ockenfels, and Christof Weinhardt, editors, *Negotiation and Market Engineering*, number 06461 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. <http://drops.dagstuhl.de/opus/volltexte/2007/1009> [date of citation: 2007-01-01].

[82] Michael Schwind, Oleg Gujo, and Tim Stockheim. Dynamic resource prices in a combinatorial grid system. In *CEC-EEE '06: 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*, page 49, Washington, DC, USA, 2006. IEEE Computer Society.

[83] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations.* Number smith1776 in History of Economic Thought Books. McMaster University Archive for the History of Economic Thought, 1776. available at http://ideas.repec.org/b/hay/hetboo/smith1776.html.

[84] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[85] Michael Strbel and Christof Weinhardt. The montreal taxonomy for electronic negotiations. In *Group Decision and Negotiation*, volume 12, pages 143–164, Washington, DC, USA, 2003. Springer Netherlands.

[86] M. Tamai, N. Shibata, K. Yasumoto, and M. Ito. Distributed market broker architecture for resource aggregation in grid computing environments. In *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 1*, pages 534–541, Washington, DC, USA, 2005. IEEE Computer Society.

[87] Paul Tucker. Market mechanisms in a programmed system, 1998.

[88] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[89] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

[90] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electionic commerce: theory and implementation. *Decis. Support Syst.*, 24(1):17–27, 1998.

[91] Edo Zurel and Noam Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 125–136. ACM, 2001.