

# An Architecture for Incorporating Decentralized Economic Models in Application Layer Networks

Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro

Computer Architecture Department, Polytechnic University of Catalonia, Spain  
{oardaiz,pchacin,ichao,felix,leandro}@ac.upc.edu

## Full affiliations:

Oscar Ardaiz  
Department of Mathematics and Informatics, Public University of Navarra  
Campus de Arrosadia, Pamplona 31006, Spain  
Phone: +34(948)168076  
oscar.ardaiz@unavarra.es

Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro  
Department of Computer Architecture, Technical University of Catalonia  
Jordi Girona, 1-3, D6; 08034 Barcelona, Spain  
Phone: +34-934017001  
pchacin@ac.upc.edu, ichao@lsi.upc.edu, felix@ac.upc.edu, leandro@ac.upc.edu

## Corresponding author:

Pablo Chacin  
Department of Computer Architecture, Technical University of Catalonia  
Jordi Girona, 1-3, D6; 08034 Barcelona, Spain  
Phone: +34-934017001  
pchacin@ac.upc.edu

**Abstract.** Efficient resource discovery and allocation is one of the challenges of any large scale Application Layer Network (ALN) such as computational Grids, Content Distribution Networks and P2P applications. In centralized approaches, the user requests can easily be matched to the most convenient resource. These approaches, however, present scalability limits in the highly dynamic and complex ALN environments. This paper, explores an architecture for incorporating fully decentralized economic mechanisms for resource allocation. These mechanisms are implemented by a set of trading agents that operate on behalf of the clients and service providers, interacting over an overlay network and interfacing with the underlying resources of the platform. A prototype of the proposed architecture is presented and the practical implications of its implementation in a grid scenario are discussed.

**Keywords:** economic algorithms, decentralization, Grid, peer-to-peer.

## 1 Introduction

The main objective of this paper is to present the on-going design and implementation of a middleware to provide the infrastructure for decentralized resource allocation markets in Application Layers Networks (ALNs), a concept which includes systems such as Grids, Peer-to-Peer (P2P) and Content Distribution Networks (CDN). The paper explains the middleware requirements, its formalization through an architecture, and its implementation using a variety of already existing middleware toolkits.

Application-layer networks (ALN) are envisioned as large, complex computer networks that allow the provisioning of services requiring a huge amount of resources. They connect large numbers of individual computers for information search, content download, parallel processing or data storage. Common concepts are Grid and Peer-to-Peer-(P2P)-Computing. Allocating and scheduling the usage of computing resources in ALNs is still an open and challenging problem [15].

Decentralized economic models are a promising approach for resource allocation in such scenarios. Its capacity to function as a coordination device among selfish participants (i.e. not requiring cooperation), and the usage of prices to synthesize information about resources, make them particularly well suited to the demands of resource allocation in ALNs, namely decentralization, scalability and information efficiency.

Even though economic models have been the object of increasing attention in the diverse research communities like grid computing, P2P and agents, the proposed architectures and frameworks lack the required flexibility because they are targeted either at a specific ALN model (e.g. P2P) or a specific market model (e.g. Auctions), and most of them rely on some form of centralized or mediating component.

The paper proposes the construction of a framework that offers a set of generic market mechanism, on which specialized strategies and policies can be dynamically plugged to adapt to specific application domains or market designs. This work has led to an on-going prototype implementation, which will provide the feedback needed to implement it in a realistic grid scenario.

The proposed architecture promotes ideas that ultimately underpin a wide range of application areas concerning the self-configuration, self-organization, self-management, self-healing, and self-protecting of computer systems, such as those envisioned in the Adaptive & Autonomic Computing research initiatives.

The rest of the paper is organized as follows. Section 2 presents the architecture, its requirements and design principles. Section 3 presents the on-going implementation of this architecture for resource allocation in a Globus GT4 platform using a set of P2P and agent toolkits. Section 4 presents the most relevant related work. Section 5 offers conclusions and planned future work.

## 2 Architecture for decentralized resource allocation

The ALN environments in which the proposed architecture must work, can be characterized as very large, dynamic in its topology, with a diverse and varying user de-

mand. These ALNs are not entirely planned, but “rise” from their own usage patterns, which in turn are frequently unpredictable due to the ad-hoc nature of the applications being used and the user communities themselves.

Any approach for resource allocation must take into account the high cost of obtaining and maintaining information about the configuration, the complexity and computational cost of optimal resource allocation decisions (due to the many parameters that must also be considered) and the need to self-adjust or adapt to environmental changes.

Therefore, the architecture of such resource allocation middleware must be evolutionary, open to changes which cannot be taken into account in the initial set-up, and able to decide, learn and adjust with limited information from neighbors.

## 2.1 Architecture requirements

The more astringent architectural requirements come from the intended adaptability to very different ALN scenarios, and the need for self-organizing components. These requirements can be summarized as follows:

- The dynamicity of the network prevents an a priori configuration of the peers or the maintenance of centralized configuration services. A peer needs to discover continuously the network characteristics and adapt accordingly
- The fully decentralized nature of the middleware demands the distribution of some critical system functions like security, resource management and topology management, without requiring specialized nodes.
- As all the system function should be implemented in all peers and they have heterogeneous properties and configurations, the system should make little assumptions about the underlying platforms and other middleware services.
- Different ALN architecture will lead to different ways to deploy the middleware components, which cannot make any assumption about the location of other components, to facilitate their (potentially dynamic) redistribution.

To deal with these architectural requirements, some fundamental design principles are established. First, economic agents are isolated from the underlying ALN characteristics, like topology, and middleware communication models, which will depend on the implementation scenarios.

Middleware should therefore offer a set of high level abstractions and mechanisms to locate and manage resources, locate other trading agents, engage agents in negotiations, learn and adapt to changing conditions. Also, the middleware should allow pluggable policies, strategies and mechanisms, which could be dynamically activated to adapt the system to different environments.

Complex behaviors are implemented by interaction of simple agents responsible for basic functions, instead of by coarse grained agents. This allows the modification of behaviors by adding new agents and by changing the interaction pattern between them.

Finally, APIs will use generic data types based on XML which can be extended or specialized on each specific environment. When possible, standard XML formats (e.g. WS-Agreements, WSRF) are used.

## 2.2 Proposed Architecture

The architecture proposed in this work is inspired by the concepts of the 'free market' economy or 'Catalaxy', first introduced by Friedrich A. von Hayek. This economics theory is distinguished by its proposal of a self-organization approach for resource allocation based on bilateral bargaining among agents with limited information [5], rather than using any form of centralized clearing mechanism to match demands and supplies, as proposed in conventional theories. The suitability of such an approach for resource allocation was assessed in the CATNET projects [1] by means of simulations, and its applicability to real application scenarios is being studied in the CATNETS project [2], within which this work is framed.

Using it as a conceptual framework, the resource allocation middleware is envisioned as a set of economic agents (representing the Client Applications, Services and Resources of the ALN) that interact between them over a P2P network, to negotiate the assignment of resources using economic criteria.

In this context the term "P2P" should be interpreted as a general approach for distributed system design, characterized by the ad-hoc nature of the system's topology and the functional symmetry of its components, which can be realized under very different architectures, ranging from unstructured and disperse networks to very hierarchical systems.

This conceptual framework is formalized in a layered architecture that consists of the following layers (see Figure 1):

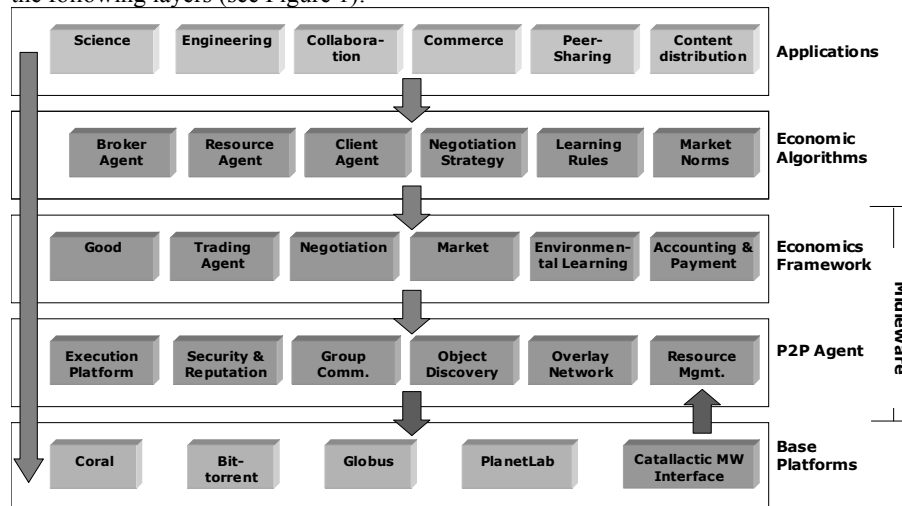


Figure 1. Architecture – Layered View

- **Applications:** is conformed by the domain specific end user applications which relay on the base platform to obtain resources, but can also offer application specific resources (i.e. scientific algorithms).
- **Economic Agents:** Implements economic algorithms that negotiate the resource allocation. These algorithms should be domain independent and platform independent. Agent's roles in this layer, as buyer, seller, broker, auctioneer, depend on the specific market being implemented.
- **Economic Framework:** offers the abstract entities, like *Trading Agents*, *Markets* and *Goods*, that support the implementation of negotiation algorithms.
- **P2P Agents:** Platform that hosts the middleware agents offering a rich development environment, covering the basic functions that will be used by all implementations; it is responsible for interfacing with the underlying base platform and complementing it when necessary.
- **Base Platform:** Supports applications and the middleware. It is (potentially) domain specific. Some components might be required to integrate this platform with the middleware.

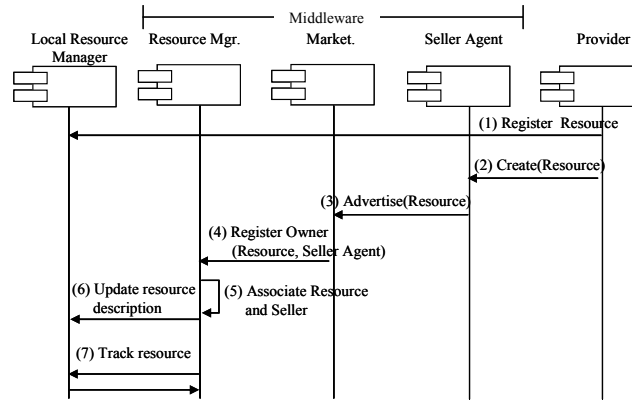
This layered architecture allows an appropriate separation between technical and economic concerns of the resource negotiation, brings a greater flexibility in the implementation of each component and promotes the independence of the economic model from the underlying platforms, which will make it more adaptable to evolving environments.

## 2.3 Detailed Design

To understand the interrelationships between the components of the architecture, it is necessary to see how they interact in different sceneries, the more relevant being the initial registry of agents, the distributed object location, which shows how the underlying P2P platform can be used to achieve a high degree of decentralization in this critical function, and the initiation of the bargaining process.

### 2.3.1 Registering resources and agents

Negotiation for resources is carried out by agents that represent the client requesting a resource and the service providers that offer that resource. How those agents are actually created is highly dependant on the scenario and the architecture of the systems requesting the resource and offering it. Figure 2 shows a generic situation.



**Figure 2.** Registering Agents and resources

After registering a resource with its local platform specific *Local Resource Manager*, the *Service Provider* instantiates a *Trading Agent* to represent it in bargaining for a specific service, which registers itself with the local *Market* agent. The *Market* uses the *Resource Manager* to associate the *Trading Agent* with the service. The *Resource Manager* can, optionally, update the resource's information in the *Local Resource Manager* to reflect, for instance, that the resource is already reserved by the middleware and cannot be offered to another application. Finally, the *Resource Manager* keeps track of the resource state (e.g. availability and usage level) and uses this information to answer queries for resources, given certain characteristics.

### 2.3.2 Resource Discovery

One of the more critical functions in the middleware is the discovery of objects like agents and resources, in a fully decentralized way, within a dynamic set of nodes (which can enter and leave the network at any time) and which must cooperate in the location of objects. Moreover, it is necessary to accommodate the discovery of different kinds of resources.

To address these requirements, the proposed architecture separates three critical functions: the *Overlay Network Agent (ONA)* manages the construction and maintenance of the logical topology of the network, keeping track of other nodes. The *Communication Agent (CA)* manages the complexities of multicasting messages over the overlay network and the *Resource Discovery Agent (RDA)* handles the search requests using a set of pluggable *Resolver Agents (RA)*, which specializes in the location of specific kinds of objects. The *ODA* is also responsible for coordinating the search with other nodes when no local information is available. Figure 3 shows how all these components interact to fulfill a search request.

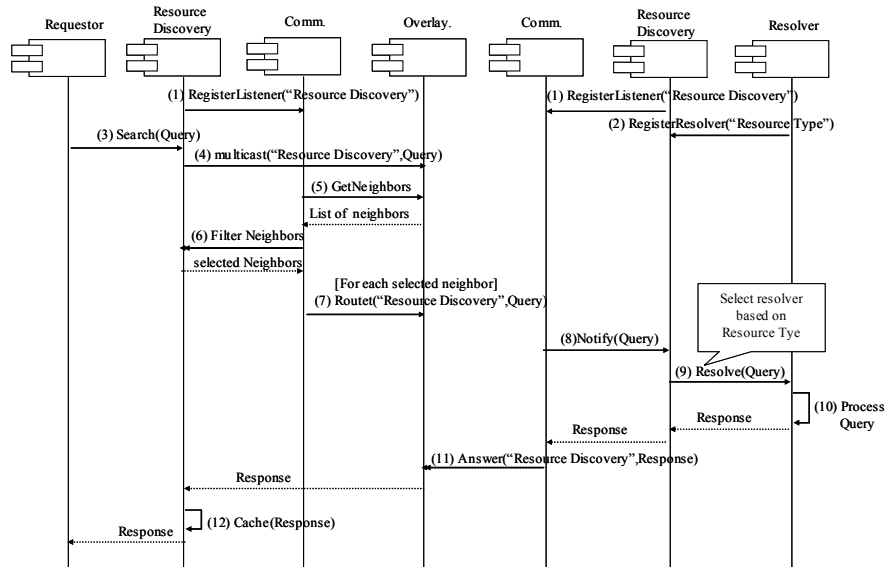


Figure 3. Fully decentralized object discovery

First, the *RDA* registers with the *CA* as a listener of multicast messages sent to the “Resource Discovery” group, and the *RA* registers with the *ODA* as a handler for queries of a specific object type. When the *ODA* receives a request, it forwards it to the corresponding local *RA* (if any) and to other remote *ODA* sending a multicast message with the query. To send the multicast, the *CA* requests a list of known nodes to the *ON* agent. Before sending the message, the *CA* agent asks the *ODA* to filter and prioritize this list, allowing the introduction of heuristics in the search (for example, based on the results from previous requests [7]). For each selected node, the *CA* agent requests the *ONA* to route the message using its knowledge of the logical network topology.

On each remote node, the *RDA* receives the multicast message and forwards it to the locally registered *RA* (if any) and returns the response to the originating node, where the *CA* collects responses from all nodes to which it sent a request. The *ODA* receives the response and sends it back to the requestor agent. It can also, optionally, store this response in a local cache to enhance performance of subsequent requests.

### 2.3.3 Negotiating for resources

The negotiation process (see figure 4) begins when a *Client Application (CA)* requests a resource from the *Broker Agent (BA)*, giving some contractual conditions (e.g. available budget) and technical specifications. How this is accomplished depends on the application scenario. The *CA* can be fully aware of the *BA* or this agent can be invoked by a component in the *CA* platform (a local resource manager, for instance). Also, the conditions and specifications can be explicitly given by the *CA*, be part of the middleware configuration or a product of the *BA* learning.

After receiving the request, the *BA* asks its local *Market Agent (MA)* for a list of potential *Seller Agents (SA)*. The *MA*, behaving as a *Resolver Agent* for the *Object Discovery Agent* (see Section 2.3.2 for a detailed description) performs a distributed search among neighbor nodes. On each neighbor node, the local *MA* requests from the *Resource Management Agent (RMA)* a list of resources that match the specifications, and their related *SAs*. Then the *MA* selects the appropriated *SA* according to the contractual conditions and sends the list back to the *MA* that started the search, which then gives this list to the *BA* after filtering it according to the contractual conditions and its experience in past negotiations. Finally, the *BA* select the *SA(s)* it wants to trade with and starts the negotiation process. The *MA* in the remote node still has the right to accept or not the negotiation based on the credentials of the *BA* and its experience in past negotiations.

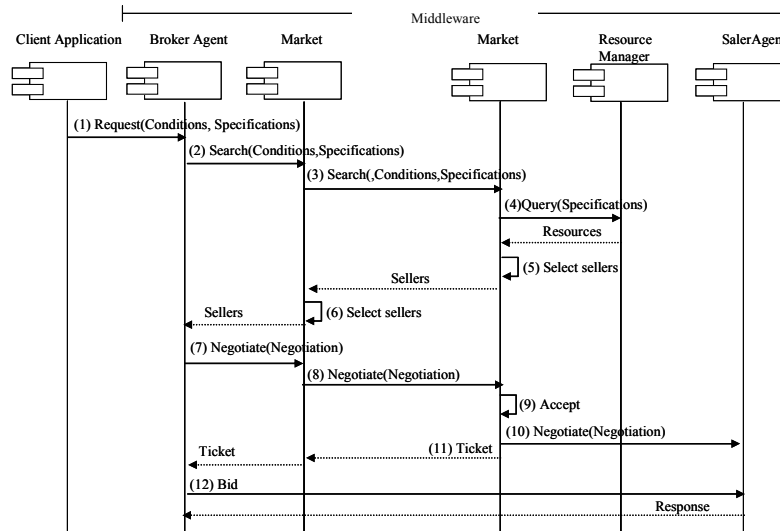


Figure 4. Negotiating for resource

### 3. Prototype

The prototype can be viewed as an early validation for the proposed architecture with a three fold objective. First, test to what extent the middleware toolkit selection consistently mapped architectural functionalities into middleware toolkit APIs. Second, to validate the feasibility of composing the middleware following the proposed separation of concerns in multiple interacting agents. Finally, to test if middleware can handle the required levels of decentralization and scalability. The results of these tests are expected to raise additional architectural requirements to be included in following iterations of the design process.

The prototype scenario considers grid resource providers offering grid services, and clients issuing simple queries to its broker agents (e.g.: “get me a service of type

x given a budget of y”). Economic agents (brokers and resource agents) negotiate on behalf of their owners for the best available deals in a decentralized, large and dynamic environment.

The prototype should have a balance between efficiency of execution and the flexibility to experiment with different implementation approaches or tools. Efficiency can be achieved by using simple and direct design approaches that take advantage of features and mechanisms optimized for the specific implementation platform, whereas the flexibility requires generic mechanism and more complex design patterns. To address this conflict in the design requirements, pluggable platform dependent components are used extensively (e.g. a resource query resolver for GT4)

Six different middleware toolkits were reviewed as potential implementation tools for the architecture: DIET [4] and JADE [8] agent platforms, J2SE [9], WSRF/OGSA [6], Web Services [17] and JXTA [10].

The analysis of middleware toolkit requirements was approached from three different viewpoints. The Functional View considers the required functionalities to implement a distributed platform according to the architectural requirements, such as scalability, decentralized information management and inter-communication capabilities. The Development View evaluates to what extent it is feasible to implement the designed architecture using the selected MW tools. Factors like API complexity, documentation and community maturity are considered

From this analysis it was found that the proposed architecture might be best implemented as a composition of different middleware toolkits, namely DIET, JXTA and WSRF/OGSA, which achieve a good balance between the functional and non functional requirements. DIET provides a modular, lightweight and scalable execution platform. JXTA offers a rich P2P networking environment and WSRF/OGSA provides full support for resource management in different scenarios. For more details on the evaluation process, see [3].

### **3.2 Prototype implementation**

The middleware is implemented as a set of simple, specialized DIET agents. The interaction of these agents is organized in a way that obeys the separation of concerns proposed by the architecture, as shown in Figure 5. Framework agents support the basic functions needed to implement economic algorithms, like access to markets. Peer Agent Layer agents implement the low level functionalities to support system execution. Service Provider agents interface with the implementation platforms (JXTA and GT4).

The Overlay Network and Communication agents, which handle the P2P communication functions, were implemented using JXTA’s Peer Resolver Protocol, which uses a DHT to maintain and route messages among Rendezvous Peers nodes [16]. The management of local resources, in this case services offered by the service providers, uses the WSRF framework offered by GT4.

This approach of separating functions as independent agents allows for scalability, since the number of agents supporting a function (e.g. resource discovery) can be dynamically adapted to workload. This separation also offers location transparency

for agents. For example, instead of engaging in complex interactions with a remote agent, a trading agent could migrate and make all the negotiation locally, returning to the node of origin with the outcome

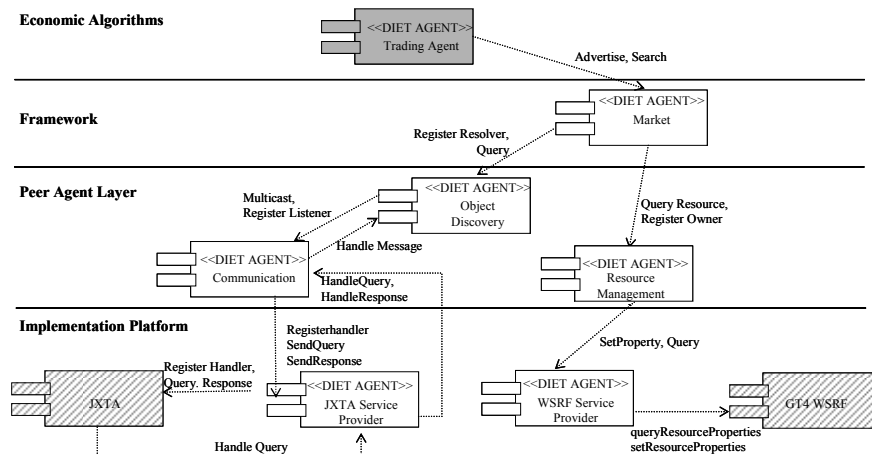


Figure 5. Organization of middleware components

### 3.3 Open issues and future steps

At present, the implementation details of the prototype presented in this paper are being worked out in order to validate the architecture. There are, however, some important issues that still need to be addressed:

- The control of low level P2P mechanisms to adapt to upper level behavior. In particular, how to control the propagation of messages based on the efficiency of object discovery process and ultimately the negotiations.
- Experimentation with alternatives to JXTA for overlay network construction including low level DHTs [11], looking for some desired properties like locality awareness and efficiency in message routing.
- Instrumentation, measurement, analysis and visualization of agent activities in a fully distributed architecture.
- A generic mechanism to activate the Broker Agent on behalf of any GT4 applications is still being designed, and is considered crucial to enable the negotiation for resources in this kind of applications without requiring modifications to the applications.

## 4. Related Work

Some recent projects, MMAPPs [13], Tycoon [12] and OCEAN [14] have proposed decentralized systems which incorporate economic models.

The MMAPPS project aimed to provide a toolkit for the development of P2P applications that uses an economic based incentive mechanism in order to coordinate and optimize these applications. MMAPPS considered that all applications and services would be developed using this framework, so integration of already existing applications was not considered (at least explicitly) in the design. In contrast, in the architecture proposed in this paper, integration into heterogeneous ALNs is a key design objective.

Perhaps the work in OCEAN [14] is the closest to the one proposed here, since their system is intended to work in a fully decentralized fashion (except the accounting mechanism), and bargaining between the agents during the negotiation process might be used. Their work differs from the work presented here in that they focus on providing matching and searching protocols and developing some resource specification and negotiation languages, while in this work the focus is on the detailed evaluation of the decentralized economic models itself.

Tycoon is a distributed market-based allocation architecture based on a local auctioning for resources on each node. Auctioneers receive fine grained requests of local resources from agents acting on behalf of applications and schedule them using efficient sealed bid auctions in a way that approximates proportional share, allowing high resource utilization rates and the adaptation to changes in demand and/or supply. One interesting feature of Tycoon is that it separates the allocation mechanism from the agents which interprets application and user preferences. This allows the specialization of agents for different applications. However, Tycoon offers no framework for the construction of those agents.

These projects confirm the trend towards self-organizing decentralized infrastructures. While the middleware used in these projects is rather tailored to the project needs, the proposed architecture should allow a middleware implementation to be used by different types of applications.

## 5. Conclusion

The proposed architecture constitutes an important step in building a middleware which will allow ALNs to operate efficiently in dynamic and heterogeneous environments. Such a middleware needs to integrate new paradigms in order to fulfill requirements such as scalability, robustness, and adaptability. This work has described how decentralized economic models could be integrated in an ALN to achieve these goals.

The proposed architecture is targeted at several application classes, from hierarchical grids to highly disperse and dynamic P2P applications, unifying what is currently available only as separate tools. To achieve this objective, the adopted architecture integrates two main application approaches: P2P and agents. The experience of implementing the prototype has shown that this combination allows for a great level of adaptability to diverse deployment scenarios and extensibility of the middleware components.

The paper has described how the proposed architecture could be implemented by combining a number of already existing middleware toolkits. This fact reveals the

lack of toolkits which fully comply with the requirements identified for dynamic and heterogeneous environments, and also demonstrates the adaptability of the proposed architecture.

This architecture thus constitutes an important step towards a formal design of middleware for resource allocation using economic algorithms.

## 6. Acknowledgements

This work was supported in part by European Union under Contract IST-FP6-003769 and the Spanish Government under Contract TIC2002-04258-C03-01.

## 7. References

- [1] CATNET Project (2003): *Catallaxy Simulation Study. Report No. D2*, [http://research.ac.upc.es/catnet/pubs/D2\\_Simulation\\_Study.pdf](http://research.ac.upc.es/catnet/pubs/D2_Simulation_Study.pdf)
- [2] CATNETS Project (2004): *Annex I – Description of work*, IST-FP6-003769
- [3] CATNETS Project (2005): *Deliverable D3.1: Selection of middleware toolkits and options for integration of catalactic mechanisms in current middleware used in peer-to-peer and grid implementations*, March 2005
- [4] *Diet Agents*, <http://diet-agents.sourceforge.net/>, visited: 10/2005.
- [5] Eymann, T.; Reinicke, M.; Ardaiz, O.; Artigas, P.; Freitag, F.; Navarro, L.: *Self-organizing resource allocation for autonomic network*, p. 656, Proceedings of 14th International Workshop on Database and Expert Systems Applications (DEXA'03), Germany, 2003.
- [6] *Globus Alliance*: <http://www.globus.org/>, visited: 10/2005.
- [7] Iammitchi, A.; Foster, I. T.: *On Fully Decentralized Resource Discovery in Grid Environments*, p.51-62, Proceedings of the Second International Workshop on Grid Computing, 2001.
- [8] *Java Agent Development Framework*, <http://jade.tilab.com/>, visited: 10/2005.
- [9] *Java Sun Developer Network*, <http://java.sun.com/>, visited: 10/2005.
- [10] *JXTA – Get Connected*, <http://www.jxta.org/>, visited: 10/2005.
- [11] Kelaskar, M.; Matossian, V.; Mehra, P.; Paul, D.; Vaidhyanathan, A.; Parashar, M.: *A Study of Discovery Mechanisms For Peer-to-Peer Applications*, Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid Workshop on Global and Peer-to-Peer on Large Scale Distributed Systems (CCGRID 2002), Berlin, Germany, IEEE Computer Society Press, pp. 444-445, 2002.
- [12] Lai, K.; Huberman, B. A.; Fine, L.; Tycoon: *A Distributed Market-based Resource Allocation System*, HP Lab, Palo Alto, Technical Report cs.DC/0404013, Apr. 2004.
- [13] MMAPPS Project (2004), *Deliverable 5: Peer-to-Peer Services Architecture*, September, 2004
- [14] Padala, P.; Harrison, C.; Pelfort, N.; Jansen, E.; Frank, M.P.; Chokkareddy, C.: *OCEAN: The Open Computation Exchange and Arbitration Network, A Market Approach to Meta computing*. p. 85, In proceedings of the International Symposium on Parallel and Distributed Computing (ISPDC'03), Oct 2003.
- [15] Snelling, D., Priol, T., et al.: *Next Generation Grid(s)*. European Grid Research 2005 - 2010. Brussels: Information Society - DG, Grids for Complex Problem Solving

- [16] Traversat, A., Abdelaziz, M.; Pouyoul, E.: *Project JXTA: Loosely-Consistent DHT Rendezvous* Walker, Sun Microsystems, Inc., [http://www.jxta.org/project/www/white\\_papers.html](http://www.jxta.org/project/www/white_papers.html), visited: 10/2005.
- [17] *Web Services*, <http://www.w3.org/2002/ws/>, visited: 10/2005.

## **Biographical notes of each author**

Oscar Ardaiz received the MS in Telecommunication Engineering from the Public University of Navarra and the Ph.D. degree in Telecommunication Engineering from the Technical University of Catalonia in 2004. He is currently working as an Assistant Professor in the Department of Mathematics and Informatics at the Public University of Navarra.

He was co-chair of the Collaborative and Learning Applications workshop (CLAG) at the CCGRID conference in 2004 and 2005. Dr. Ardaiz's research interests include design and evaluation of distributed systems and collaborative applications.

Pablo Chacin is a PhD student at the Technical University of Catalonia (UPC). He is currently working in the CATNETS EU research project. Pablo's research interests include economic inspired distributed resource allocation mechanisms and architectures for Grid and peer-to-peer systems.

Isaac Chao is a PhD student at the Technical University of Catalonia (UPC). He is currently working in the CATNETS EU research project. Isaac's research interests include economic inspired distributed resource allocation mechanisms and reputation mechanisms.

Felix Freitag received the MS in Electrical Engineering from the Technical University of Munich and the Ph.D. degree in Telecommunication Engineering from the Technical University of Catalonia in 1998.

He is currently working as an Assistant Professor in the Department of Computer Architecture at the Technical University of Catalonia. Dr. Freitag's research interests include performance evaluation of distributed and parallel systems.

Leandro Navarro joined the Computer Architecture Department of UPC as an associate professor in 1988, after receiving his graduate degree on Telecommunication Engineering from UPC.

He received his Ph.D. from UPC in 1992. Since 1985, he is working at the Computer Architecture Department (Departament d'Arquitectura de Computadors).

Leandro's research interests include the design of scalable and cooperative Internet services and applications.

## **Figure captions**

- Figure 1. Architecture – Layered View
- Figure 2. Registering Agents and resources
- Figure 3. Fully decentralized object discovery
- Figure 4. Negotiating for resource
- Figure 5. Organization of middleware components