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.es

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

1 Introduction

The main objective of this paper is to present the ongoing design and implementation of middleware architecture for the implementation of decentralized resource allocation markets in various Application Layers Networks (ALNs), concept involving systems such as Grids, Peer-to-Peer (P2P) and Content Distribution Networks (CDN). The elicitation of middleware requirements, its formalization through middleware architecture and the evaluation carried out over middleware toolkits suitable for its implementation are explained.

The more distinctive characteristics of the architecture presented in this paper are the usage of economics models as the conceptual framework that guided its design, and its proposed integrability to heterogeneous ALNs.

This work had lead to an ongoing prototype implementation, which will provide feedback for new design decisions and lead to a stable "proof of concept" prototype which will be used to fully test these concepts in a realistic scenario.

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

1.1 Catallactic approach to resource allocation

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 [SnPr03]

The CATNETS [Catn04] project investigates a 'free market' economic self-organization approach, the 'Catallaxy' by Friedrich A. von Hayek, as the basis for self-organizing resource allocation in ALNs. A preliminary evaluation of 'Catallactic' mechanisms in the FET assessment project CATNET [Catn03] by simulation has shown positive results.

CATNETS promotes ideas that ultimately underpin a wide range of application areas, not only in the Next Generation Grid [SnPr03], but also in using self-configuring, self-healing, self-organizing and self-protecting computer systems like envisioned in the upcoming Adaptive & Autonomic Computing research initiatives [IBM01]. The long term objective is here to introduce economic concepts (alternatively to existing biological concepts) for the creation of on-demand e-business environments, Grids, peer-to-peer networks, web service environments, ad-hoc sensor networks, etc.

The rationale for selecting scenarios in ALN where Catallaxy has potential for success can be summarized as follows:

- Dynamic: changing environments and the need for adaptation to changes is one of the potential areas where Catallaxy can have a competitive advantage.
- Diverse: requests may have different priorities and responses should be assigned according to them.
- Large: with such number of elements that locality is required to scale
- Partial knowledge: it is not possible to know everything on time because of its high cost. This can be caused by scale issues such as a large number of elements, number of messages, or communication latency.
- Complex: many parameters must be taken into account, many messages. Learning mechanisms are necessary to self-adjust or adapt to changes, and optimal solutions are not easily computable.
- Evolutionary: open to changes which cannot be take into account in the initial set-up, and able to learn and decide with limited information: neighbors, few parameters that are summarized as a single price value, few historic data, etc.

1.2 Related work

Even when the usage of economic based mechanism for resource allocation is far from being new, to the best of our knowledge, there is no single project that attempted to develop a general architecture for a broad spectrum of ALNs.

MMAPPS project [Mmap04] aimed to provide a toolkit for the development of P2P applications that uses economic based incentive mechanism that allows the coordination and optimization of 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, integrability to heterogeneous ALNs is a key design objective.

Also the work in OCEAN [PHP+03] is close to the one proposed here, since their system is intended to work in a fully decentralized manner (except or the accounting mechanism), and bargaining between agents during negotiation process might be used. Their work differ from the work presented here in that they focus on providing matching and searching protocols and developing some resource specification an negotiation languages, while in this work the focus is in detailed evaluation of the decentralized economic models itself.

2 Architecture

To achieve the astringent requirements imposed by diversity of potential implementation scenarios, a layered architecture have been proposed that allows a clear separation of design concerns, facilitates the division of design and development work, and gives the freedom to experiment with different implementation options, but at the same time avoiding the risk of incompatibilities.

The architecture also identifies the key architectural requirements that guided the evaluation of potential implementation options, including the adoption of already existing middleware toolkits systems and development platforms

Finally, the importance of the architecture goes far beyond the simple documentation of technical elements. Architecture serves as “the blueprint for both the system and the project developing it” and therefore it helps in the definition of how the work can be organized.

2.1 Architecture requirements

The more astringent requirements come from the need of for self-organization and adaptability to very different ALN scenarios. These requirements can be summarized as follows:

- The key requirement is the efficient resource allocation in very large ALNs, with thousands of nodes in a highly dynamic environment, where nodes enter and leaves frequently.

- 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 catallaxy approach demands the distribution of some critical system functions like security, resource management, 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 P2P system should make little assumptions about the underlying platforms.
- 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 the architectural requirements, some fundamental design principles were established. First, economic agents will be isolated from the underlying ALN characteristics, like topology, and middleware's 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 will be implemented by interaction of simple agents responsible for basic functions, instead of by coarse grained agents. This will allow the modification of behaviors by adding new agents and changing the interaction pattern between them.

Finally, APIs will use opaque data types which can be extended or specialized on each specific implementation. These opaque data types can by language dependent, like abstract parameter objects in Java or can be language neutral, like XML documents.

2.2 Proposed Architecture

To address the architectural requirements defined in the previous sections, a layered architecture was proposed (see figure 1), which allow the separation of the different concerns to manage them individually without missing the coherence of the architecture as a whole. The layers identified in the architecture are:

- **Applications:** is conformed by the domain specific end user applications which relay on the base platform to obtain resources, but also can 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, depends 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 Catallactic middleware. It is (potentially) domain specific.

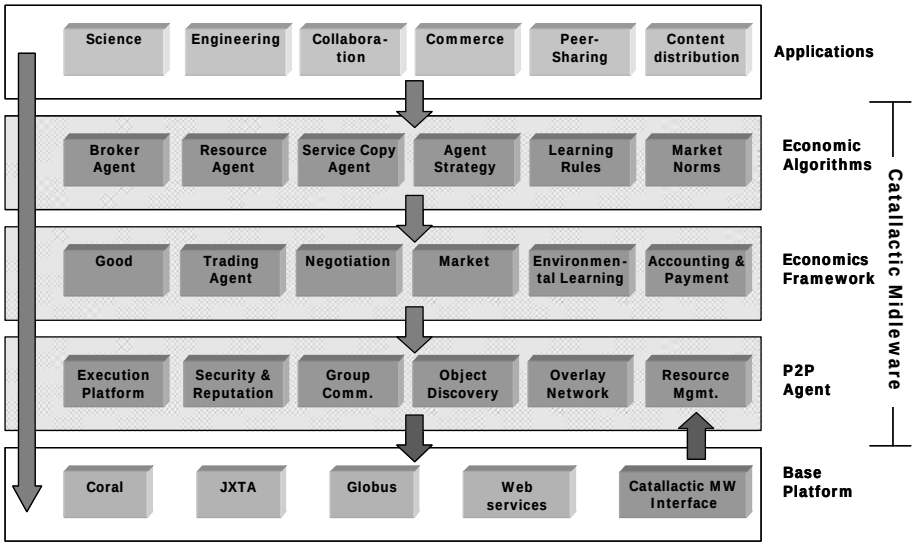

Figure 1 Architecture – Layered View

2.3 Detailed Design

To understand the interrelationships between the components of the architecture, it is necessary to see how they interact in different scenarios, being the more relevant: the initial registry of resources in markets, the distributed resource search, and the bargaining process.

2.3.1 Registering resources and agents

Negotiation for resources is carried out by *Broker Agents* representing the client requesting a resource, and *Resource Agents* representing resource providers. How those agents are actually created is very dependant on the scenario and the architecture of the systems requesting the resource and offering it. Figure 4 shows a generic situation.

The *Service Provider* is responsible for creating and registering a resource with its platform specific *Local Resource Manager*. It then instantiates a *Resource Agent*,

which registers itself to the *Market Agent* to handle negotiations for that resource. The *Market Agent* uses the *Resource Manager Agent (RMA)* to associate the *Resource Agent* with the specific resource. The *RMA* 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 other application. Finally, the *RMA* keeps track of the resource state (e.g. availability and usage level) and uses this information to answer queries for resources given a certain characteristics.

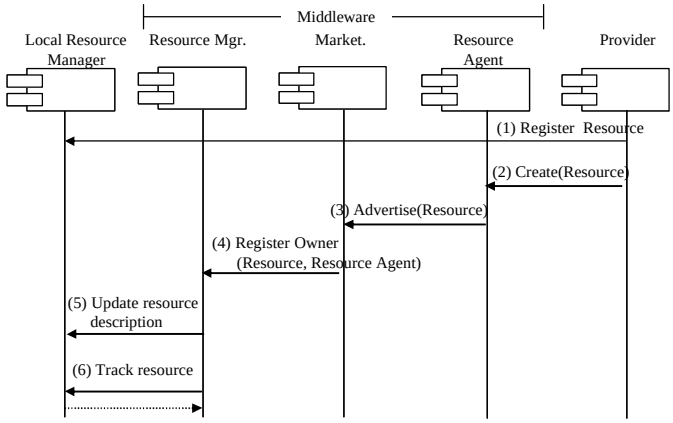

Figure 2. Registering Agents and resources

2.3.2 Resource Discovery

One of the more critical functions in the middleware is the discovery of resources in a fully decentralized way, within a dynamic set of nodes (that can enter and leave the network at any time) which must cooperate in the location of objects.

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. *Communication Agent* (*CA*) manages the complexities of multicasting messages over the overlay network and the *Object Discovery* Agent (*ODA*) handles the search requests using a set of pluggable *Query Resolver Agents* (Q*RA*), which specializes in the location of a specific kind of objects. *ODA* is also responsible to coordinate the search with other nodes when no local information is available. Figure 3 shows how all these components interact to fulfill a search request.

First, *ODA* registers to the *CA* as a listener of multicast messages send to the "ObjectDiscovery" group; the *QRA* register to the *ODA* as a handler for queries of a specific object type. When the *ODA* receives a request, forwards it to the corresponding local *QRA* (if any) and to other remote *ODA* sending a multicast message with the query.

Before sending 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 priori-

tize this list, allowing the introduction of heuristics in the search (for example, based on the results from previous requests [IaFo01]). For each selected node, *CA* agent requests the *ONA* to route the message using its knowledge of the logical network's topology.

On each node, the *ODA* receives the multicasted message and forwards to the locally registered *RA* (if any) and returns the response to the originating node, where the *ODA* sends back to the requestor. It can also, optionally, store this response in a local cache to enhance performance of subsequent requests.

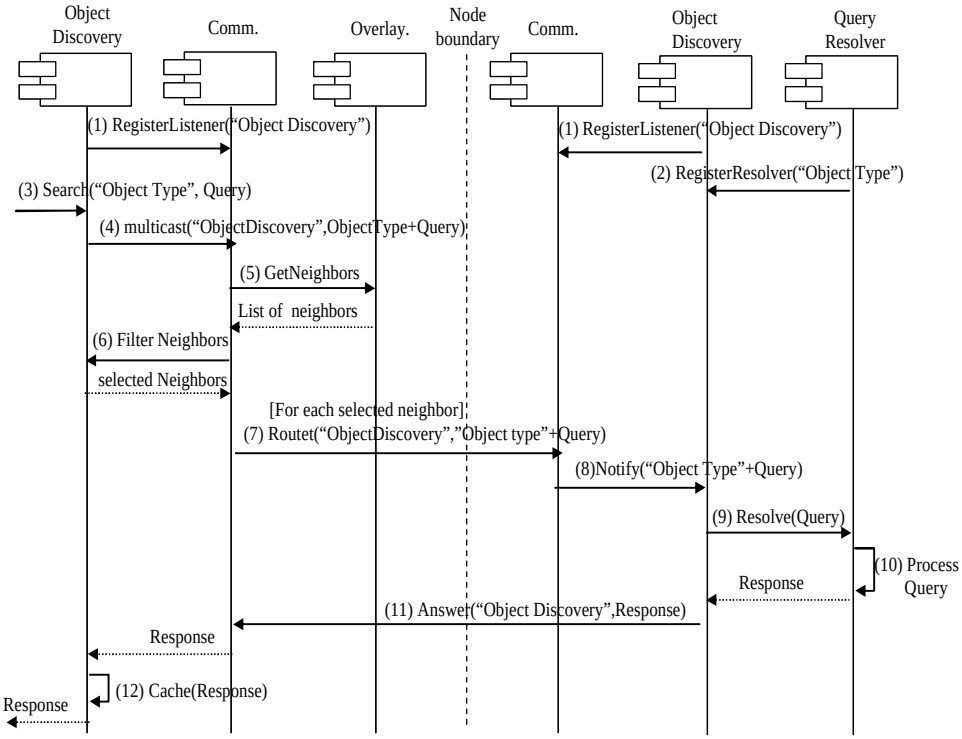

Figure 3 Fully decentralized object discovery

2.43.3 Negotiating for resources

Negotiation process begins when a *Client Application* (*CA*) request a resource to the *Broker Agent (BA),* giving some contractual conditions (e.g. available budget) and technical specifications. Figure 4 shows this process.

How the *CA* communicates with the *BA* 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's* platform (a local resource manager, for instance. Also, the conditions and specifications can be explicitly given by the *CA* to the *BA*, be part of the middleware's configuration parameters or a result of the *BA* learning process.

After receiving the request, the *BA* asks the *Market Agent* (*MA*) for a list of potential *Resource Agents* (*RA*). The *MA* performs a distributed search (see section

2.3.2 for a detailed description) to find the resources that match the specifications and the *RAs* that handle the negotiation for those resources. Then the *MA* filters the *SA* according to the contractual conditions. The *BA* selects from the given list the *RA*(s) it want to trade with (based on previous experience, for instance) and starts the negotiation process. The *MA* has the right to allow or not the negotiation based on rules governing the market.

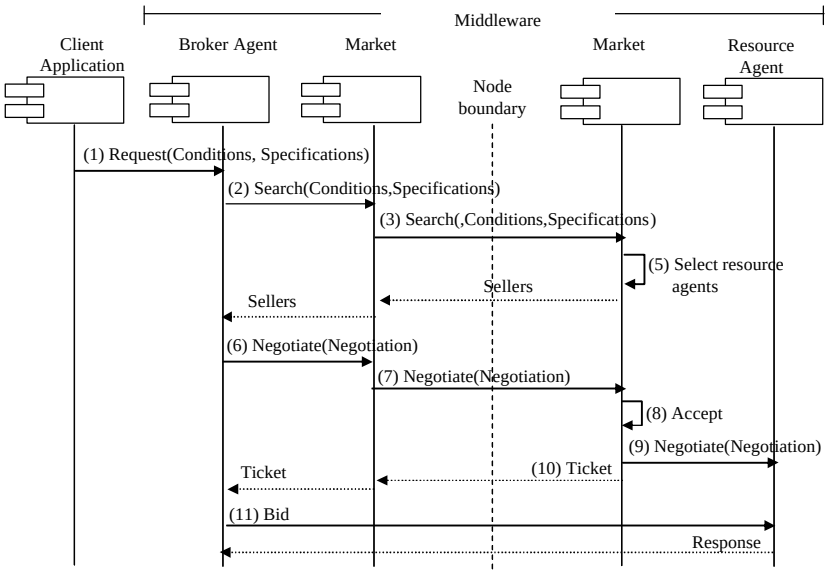

Figure 4 Negotiating for resource

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 evaluation (described in section 3.1 “Middleware toolkit selection”) consistently mapped architectural functionalities into middleware toolkits APIs. Second, validate the feasibility to compose the middleware following the proposed separation of concerns in multiple interacting agents. Finally, allow to test in 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 scope of this prototype was limited to the proposed middleware architecture. Implementation of the full catallactic economic model or applications was not attempted; The prototype scenario consider a 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 in execution and the flexibility to experiment with different implementation approaches or tools. Efficiency can be achieved by using simple and direct designs approaches that takes advantage of features and mechanisms optimized for the specific implementation platform, whereas the flexibility requires generic mechanism and more complex designs patterns.

3.1 Middleware toolkit selection

The analysis of middleware toolkits requirements was approached from three different views. Functional view considers the required functionalities to implement a distributed platform according to the architectural requirements presented in section 2.3 “P2P agent Layer”. Technical View reflects non-functional requirements such us scalability, decentralized information management and inter-communication capabilities. Development View evaluates to what extent is feasible to implement the designed architecture using the selected MW tools. Factors like API complexity, documentation and community maturity are considered

Six different middleware toolkits were reviewed as potential implementation tools for the architecture: DIET [Diet05] and JADE [Jade05] agent platforms, J2SE [J2se05], WSRF/OGSA [Glob05], Web Services [WeSe05] and JXTA [Jxta05]. For greater detail on the evaluation process see [Catn05]

A condensed view of all requirements, in functional, technical and development views is obtained considering the following criteria:

- **Modularity:** architectural flexibility required to implement the Catallactic middleware into different platforms and using diverse middleware toolkits
- **Amenability**: ability to adapt as much as possible of the ALN domains, like Grid, P2P and CDN
- Performance & Scalability: allow the organization of a huge number of software agents in a decentralized way, and their interactions.
- **Completeness:** The set of functionalities provided by the middleware toolkit should allow covering as much as possible of the desired requirements of the P2P Agent Layer).
- **Development:** is mature and offers rich set of development tools and good documentation.

Figure 5 illustrates this unified view. Each of the pentagon axis represents one of the criteria. For each criteria the two middleware toolkits best covering it are indicated.

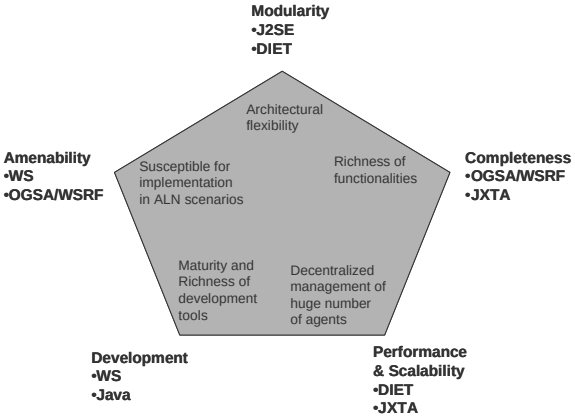

Figure 5 Key middleware toolkit evaluation criteria

From the Figure 5 above, it can be seen that the proposed middleware 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.

3.2 Prototype implementation

The middleware was implemented as a set of simple, specialized DIET agents. The interaction of these agents is organized in a way that resembles the separation of concerns proposed by the architecture, as is shown in figure 6.

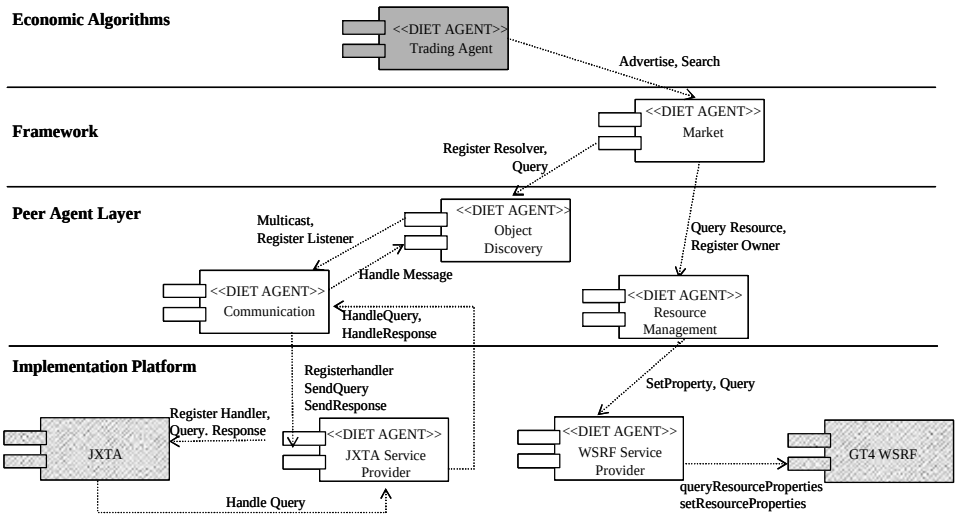

Figure 6 Organization of middleware components

Framework agents supports 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, Object Discovery and Communication functions where implemented using JXTA Peer Resolver Protocol in a network of Rendezvous Peers that uses a DHT to maintain and route messages among nodes [TAP03]. The management of local resources, in this case services offered by the service providers, uses the WSRF framework offered by GT4.

In the proposed implementation, Trading Agents implementing economic algorithms share the supporting agents from lower levels. This separation allows for scalability as the number of supporting agents can be dynamically adapted to workload. This separation also offers location transparency for agents.

It has been quite straightforward to implement interfaces between DIET Agent and JXTA and GT4, provided DIET don't have default services for P2P discovery or Resource Management, but rather aims to be modular enough to allow them to be plugged by developers. A *Service Provider* pattern has being extensively used, with specialized agents interfacing trading agents to basic JXTA and GT4 functionalities.

The mapping of overlay network, object discovery and group communication agents into JXTA middleware toolkit can be viewed as an achievement of architectural flexibility, considering JXTA organizes these functions in very different manner than proposed. This mapping, however, was not without limitations, as is dicussed in the next section.

4. Conclusions

The proposed architecture brings a set of important benefits, namely an appropriated separation of concerns, a great deal of flexibility in the arrangement of components and a strong "agnosticism" regarding the underlying platforms, that will facilitate its adaptation to diverse ALN scenarios.

In particular, the separation of the economic agents from the basic mechanism of the middleware allows the implementation of diverse market models on top of a single middleware infrastructure.

In the same line of thought, the separation of functions in the P2P Agent Layer offers a significant degree of flexibility when implementing the search algorithms, allowing each of the components to be replaced or adapted to specific ANL's requirements. For example, using pluggable resolvers allows the utilization of native platform's resource discovery mechanism, like the Globus's, MDS Index, as was shown in section 3 "Prototype".

However, during the architecture design process we have identified some critical issues that must be addressed when developing an open middleware intended to fit in diverse implementation architectures:

- The lack of standards for APIs for some critical functions like P2P overlay management could limit the experimentation with different middleware toolkits. This might be overcome developing a set of abstract APIs and map them to each implementation, but the risk is to find discrepancies in the semantic or to end up with functions with a semantic so generic that results unintelligible.
- To implement resource allocation polices in diverse ALNs, the middleware needs a flexible framework that allows a consistent view and management of resources using a uniform set of mechanisms, independently of the how each base platform handles the allocation monitoring of its resources. This brings some important consideration about interaction patterns between agents and the base platform and also the mapping from generic economic parameters (e.g. price) and the underlying technical parameters in the base platform.
- Experimentation will be important for to test critical features that might have a significant impact on the architecture. Models for analysis and interpretation of results from complex interaction between system components will be needed to obtain some useful insight [EdBr04].

This prototype will provide additional insight into technical and functional properties of the architecture which will be used to confirm the feasibility of its implementation. The results from experiments with this prototype will reveal implementation issues and provide a framework to evaluate different design alternatives.

4.1 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 JXTA mechanisms. In particular, how the JXTA's Resolver Service controls the propagation of messages (which has a great impact to the performance and the efficiency of the object discovery process) and the construction of the Rendezvous Peer View [TAP03] to introduce heuristics in search, based in previous query processing. We also plan to experiment with alternatives to JXTA for overlay network construction including low level DHTs [KMM+02], looking for some desired properties like locality awareness and efficiency in message routing.
- A generic mechanism to activate the Broker Agent on behalf of any Globus applications is still being designed and is considered crucial to make these applications Catalaxy-enable.
- Implementation of instrumentation, measurement, analysis and visualization of agent's behavior in a fully distributed architecture.

References

[Catn03] CATNET Project (2003), "Catallaxy Simulation Study. Report No. D2", http://research.ac.upc.es/catnet/pubs/D2_Simulation_Study.pdf

[Catn04] CATNETS Project (2004), "Annex I – Description of work", IST-FP6-003769

[Catn05] CATNETS Project (2005): "Deliverable D3.1: Selection of middleware toolkits and options for integration of catallactic mechanisms in current middleware used in peer-to-peer and grid implementations, March 2005

[Diet05] http://diet-agents.sourceforge.net/Index.html

[EAB+99] F. Eliassen, A. Andersen, G.S. Blair, F. Costa, G. Coulson, V. Goebel, O. Hansen, T. Kristensen, T. Plagemann, H.O. Rafaelsen, K.B. Saikoski, Y. Weihai Yu, (1999), "Next generation middleware: requirements, architecture, and prototypes", Proceedings. 7th IEEE Workshop on Future Trends of Distributed Computing Systems, Cape Town , South Africa, 1999

[EdBr04] Edmonds, B. and Bryson, J. (2004) The Insufficiency of Formal Design Methods - the necessity of an experimental approach. In: The Third International Joint Conference on Autonomous Agents and Mutli-Agent Systems

[ERA+03] T. Eymann, M. Reinickke, O. Ardaiz, P. Artigas, F. Freitag, L. Navarro (2003), "Self-organizing resource allocation for autonomic network", Proceedings. 14th International Workshop on Database and Expert Systems Applications, Germany, 656- 660

[Glob05] http://www.globus.org/

[IaFo01] A. Iamnitchi , I. T. Foster (2001), "On Fully Decentralized Resource Discovery in Grid Environments", Proceedings of the Second International Workshop on Grid Computing, p.51-62

[IBM01] IBM Corp.: Autonomic Computing. Yorktown Heights, NY: IBM 2001,

[J2se05] http://java.sun.com/

[Jade05] http://jade.tilab.com/

[Jxta05] http://www.jxta.org/

[KMM+02] M. Kelaskar, V. Matossian, P. Mehra, D. Paul, A. Vaidhyanathan and M. Parashar, 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,, Berlin, Germany, IEEE Computer Society Press, pp. 444-445 May 2002.

[Mmap04] MMAPPS Project (2004), "Deliverable 5: Peer-to-Peer Services Architecture", September, 2004

[PHP+03] Pradeep Padala, Cyrus Harrison, Nicholas Pelfort, Erwin Jansen, Michael P Frank and Chaitanya Chokkareddy. OCEAN: The Open Computation Exchange and Arbitration Network, A Market Approach to Meta computing. In proceedings of the International Symposium on Parallel and Distributed Computing (ISPDC'03), Oct 2003

[SnPr03] Snelling, D., Priol, T., et al.: Next Generation Grid(s). European Grid Research 2005 -2010. Brussels: Information Society - DG, Grids for Complex Problem Solving

[TAP03] Traversat, M. Abdelaziz, and E. Pouyoul, Project JXTA: Loosely-Consistent DHT Rendezvous Walker, Sun Microsystems, Inc., http://www.jxta.org/project/www/docs/jxtadht.

[WeSe05] http://www.w3.org/2002/ws/