Integration of Decentralized Economic Models for Resource Self-Management in Application Layer Networks[*]

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

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

Abstract. Resource allocation is one of the challenges for self-management of large scale distributed applications running in a dynamic and heterogeneous environment. Considering Application Layer Networks (ALN) as a general term for such applications including computational Grids, Content Distribution Networks and P2P applications, the characteristics of the ALNs and the environment preclude an efficient resource allocation by a central instance. The approach we propose integrates ideas from decentralized economic models into the architecture of a resource allocation middleware, which allows the scalability towards the participant number and the robustness in very dynamic environments. At the same time, the pursuit of the participants for their individual goals should benefit the global optimization of the application. In this work, we describe the components of this middleware architecture and introduce an ongoing prototype.

Keywords: Resource Allocation, Autonomic Systems, Decentralized Economic Models, Middleware Architecture

1 Introduction

"Autonomic Communication is a paradigm in which the applications and the services are not ported onto a pre-existing network, but where the network itself grows out of the applications and the services that end users wants" [ACCA04].

Under this vision, large scale Application Layers Networks (ALNs), including computational Grid, Peer-to-Peer and Content Distribution Networks, are evolving towards the notion of "Selfware", which achieves local autonomic control and global self-organization applying management policies in a decentralized way. One of these key polices is the assignment of resources to ALN's services.

Within such dynamic and heterogeneous environments, centralized allocation instances are limited in performing an efficient resource allocation task. To operate in such environments, the decision making processes within the application needs to be transferred to decentralized components with autonomic behavior.

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

We propose a resource allocation middleware architecture which facilitates the application of resource management in a decentralized, autonomous and infrastructure independent way. It offers a generic decentralized negotiation framework, on which specialized negotiation strategies and policies can be dynamically plugged to adapt to specific application domains and market designs.

This middleware's architecture is based on the ideas of the decentralized economic model known in the economic community as “Catallaxy”, on which a state of coordinated actions, the "spontaneous order”, comes into existence through the bartering and communicating of economic agents with posses only partial knowledge of the market participants and price's evolution.

The rest of this paper is organized as follows. Section 2 presents requirements for resource allocation in ALNs, exploring the characteristics of this kind of distributed applications, the issues related to resource self-management and the applicability of decentralized economic models to address those requirements. Section 3 describes the proposed middleware architecture, presenting its design principles and how the components interact to address resource allocation requests. Section 4 presents the related work. Finally, section 5 present our conclusions and proposes some future work.

2 Resource self-management in ALNs

Application-layer networks (ALN) such as Grid, Peer-to-Peer (P2P) and Content Distribution Networks (CDN) are envisioned as large-scale distributed applications that allow the provisioning of services using the needed resources from a large, heterogeneous and dynamic resource pool. However, allocating and scheduling the usage of computing resources in ALNs is still an open and challenging problem.

In this section we introduce the characteristics of the targeted ALNs, the specific requirements for resource allocation and the principles of decentralized economic mechanisms that allow an efficient resource allocation in this kind of environments.

2.1 Characteristics of Large-scale Application Layer Networks

Applications that are targeted have the following common characteristics:

- Dynamic: changing environments and the need for adaptation.
- Large: having such number of elements that locality is required in order to scale
- Partial knowledge: it is not possible to know everything in time. This can be caused by scale issues such as a large number of elements, number of messages, or communication latency.
- Evolutionary: open to changes which cannot be taken into account in the initial set-up.
- Diverse: requests may have different priorities and responses should be accordingly assigned.

- Complex: many parameters must be taken into account to take decisions. Learning mechanisms are necessary to self-adjust or adapt to changes, and optimal solutions are not easily computable.

In order to identify the application classes, we map the parameter space into two dimensions. We consider *Configuration Complexity*, which includes the dynamics of the configuration, lack of global knowledge and evolutionary environment and *Allocation Complexity*, which includes the diversity of requirements and complexity of allocation demands. Figure 1 shows a two-dimensional map with an approximate location of three important application classes.

It can be seen in Figure 1 that that our target application space is situated in the upper right area of the diagram. In our view, none of the three application classes do fully exploit this space, but we expect that distributed applications still to come are aimed to work in this environment. This fact emphasizes the need for a description of a software architecture which integrates decentralized components.

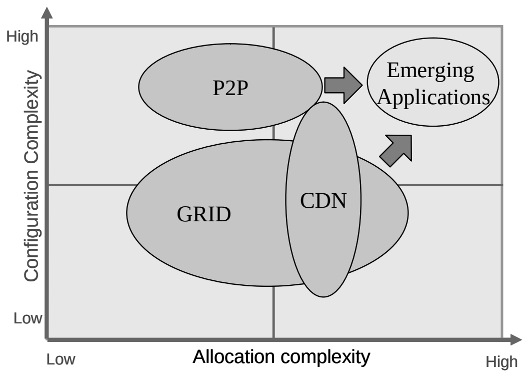

Figure 1. Target application space

Within such environment, applications with a centralized allocation instances are limited in performing an efficient resource allocation task. To operate in such environments, the decision making processes within the application needs to be transferred to decentralized components with autonomic behavior.

2.2 Resource Allocation and Self-Management in ALNs

We expect ALN to be built from basic services that can be dynamically combined to form value-added complex services. These basic services require a set of resources, which need to be co-allocated to provide the necessary computing power.

Therefore, the introduction of new services into this kind of networks, due to the dynamic nature of the environment, precludes any manual or static configuration and demands a self-organization approach, where services should be able to self-configuration, self-optimization and self-healing [WHW+04].

One goal of self-managed network services is to move away from individual system configuration management to policy management. This approach brings a higher level of abstraction to management by introducing a policy from which the configuration is derived, allowing components of the infrastructure to apply these derived configurations to the individual systems across the environment.

In this context, self-managing service's resources involves defining SLA policies for services and resources, mapping required SLA to resources needs, discovering resources that guarantee an adequate QoS, allocating resources ensuring that allocation policies are meet and providing a management interface to monitor an control service life-cycle. Because of the dynamicity of the environment we envision, the service allocation framework must address some specific issues:

- Situateness: services must be aware of its location and the closeness of peer services to collaborate
- Dynamic (re)configuration: usage patters from service users are unpredictable, therefore neither the location nor the number of service instances could be known in advance. New instances must be created and located as needed
- Topology neutrality: services deployed in the ALN could have very different interaction topologies. Some will be structured in a rather hierarchical overlay, like content distribution, while other interact in a closely connected P2P overlay.
- Autonomy: service and the resources it uses will span multiple administrative domains so each of them should be allowed to take decisions autonomously.

We propose a resource allocation middleware architecture based on decentralized economic models, which facilitates the application of resource management polices according to the above requirements (i.e. in a decentralized, autonomous and infrastructure independent way).

This resource allocation middleware has been envisioned as a set of economic agents (representing the Client Applications, Services and Resources of the ALN) that interact between them and with the software components of the underlying ALN, to coordinate, in a decentralized way and using economic criteria, the assignment of resources, as can be seen in the Figure 2.

Direct agent to agent bargaining allows participants to use thee negotiation strategy more suitable to its objectives and current circumstances. Local bilateral bargaining also facilitates the scalability of the system and the quick adaptation to local fluctuations in resource allocation dynamics.

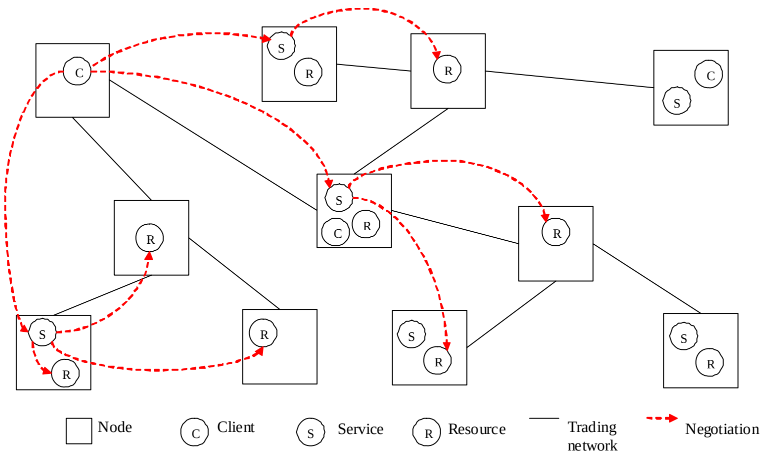

Figure 2. Decentralized allocation of resources in an ALN.

2.3 Decentralized economic models for resource allocation

The decentralized economic models applied in our work are based on the ideas of the 'free market' economy, the 'Catallaxy' proposed by Friedrich A. von Hayek, as a self-organization approach for information systems [EyPa00]. It is opposed to “plan economy” where a central entity has global knowledge of the system and commands every entity decisions. In Catallaxy, in fact, a central presumption is "constitutional ignorance”, assuming that it is impossible have global knowledge.

The Catallaxy concept bases on the explicit assumption of self-interested participants who try to maximize their own utility and choose their actions under incomplete information and bounded rationality. Agents subjectively weigh and choose preferred alternatives, and communicate using commonly accessible markets, where they barter about access to resources held by other participants. The market here is nothing more than a communication bus – it is not a central entity of its own and does not participate in matching participants’ requirements using some optimization mechanisms.

The goal of Catallaxy is to arrive at a state of coordinated actions, the "spontaneous order”, which comes into existence through the bartering and communicating of the community members with each other and thus, achieving a community goal that no single user has planned for. It promotes ideas that ultimately underpin self-configuring, self-healing, self-organizing and self-protecting computer systems like envisioned in the Adaptive & Autonomic Computing [IBM01] and Autonomic Communication [ACCA04] research initiatives.

The applicability of this approach for resource allocation in the context of ALNs has been evaluated in simulation studies which shown it is particularly well suited to handle highly dynamic environments [Catn03]. We address the task to develop a middleware architecture that helps to embody this concept in diverse applications domains.

3 Architecture

We believe the requirements imposed by the application scenarios analyzed demand an innovative approach for the construction of the resource allocation middleware. The proposed approach is the construction of a framework that offers a set of generic negotiation mechanism, on which specialized strategies and policies can be dynamically plugged to adapt to specific application domains or market designs. The 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. We will first analyze the architectural requirements that need to be addressed to fulfill this vision and then present the proposed architecture.

3.1 Architecture requirements

The more astringent architectural requirements come from the need for self-organization and adaptability to very different ALN scenarios. 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 approach requires 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.
- Given the multi-service nature of today’s ALNs, one important goal of the architecture is to allow the coexistence of diverse specialized market models on top of a single middleware infrastructure.
- The middleware should allow pluggable policies, strategies and mechanisms, which could be dynamically activated to adapt the system to different environments.

3.2 Proposed Architecture

We propose a layered architecture shown in the figure 3. This layered approach offers the palpable benefic of a clear separation of concerns between the layers, which beside helping in tackling the complexity of the system, also facilitate the construction of a more adaptable system as the upper layers can be progressively specialized (by means of pluggable rules and strategies) into specific application domains.

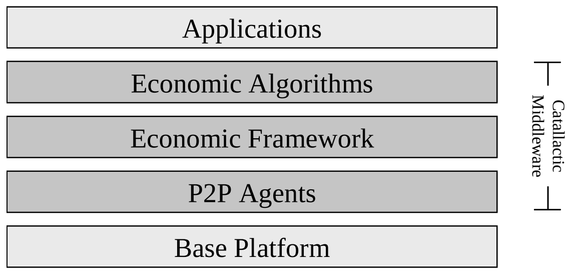

Figure 3. A layered architecture for resource allocation

Agents in the *Economic Algorithms Layer* are responsible for implementing the high level economic behavior contained in the economic algorithms layer (negotiation, learning, adaptation to environment signals, other agent's strategies and its own outcomes). Applications themselves do not participate (and are not actually aware of) the negotiation, but delegate it to the economic agents.

Economic agents rely on a lower level layer, the *P2P Agent Layer*, for the self-organization of the systems and the interaction with the base platform that ultimately manages the resources being traded. This layer offers key functions like the maintenance of the trading network topology following a P2P paradigm, the decentralized resource discovery and the group communication among agents.

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.

Between those two layers, a *Framework Layer* isolates economic agents from technical complexities; much in the same tenor that modern online trading platform allows non expert users to trade stocks. This framework offers basic functions like searching for suitable providers given a resource specification, handle the exchange of messages during the negotiation process, keeping track of the evolution of the negotiation for further adaptation of strategies.

3.3 Dynamic view

To appreciate 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 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.

3.3.1 Registering resources and agents

Negotiation for resources is carried out by agents that represent the client requesting a resource and the providers that offers that resource. How those agents are actually created is very dependant on the architecture of the systems requesting the resource and offering it. Figure 4 shows a generic situation.

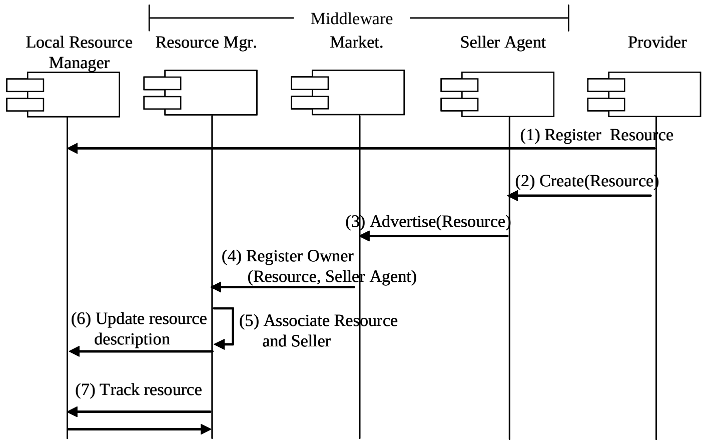

Figure 4. Registering Agents and resources.

The Resource *Provider* application, registers a resource with its local platform specific *Local Resource Manager* (which is part of the execution platform and outside the middleware), and instantiates a *Seller Agent (SA)* to represent it in bargaining for a specific service. The *SA* registers itself to the local *Market* agent, which uses the middleware's *Resource Manager Agent (RMA)* to associate the *SA* with the 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.

3.3.2 Negotiating for resources

Negotiation process begins when a *Client Application* (*CA*) requests a resource to 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 invoke directly the *BA* or it can be invoked by a component in the *CA's* platform (a local resource manager, for instance) in response for a request for resources. Also, the conditions and specifications can be explicitly given by the *CA*, be part of the middleware configuration or a result of the *BA* learning during past negotiations.

After receiving the request, the *BA* asks its local *Market Agent* (*MA*) for a list of potential *Seller Agents* (*SA*). The *MA* performs a distributed search among neighbor nodes. On each neighbor node, the local *MA* requests 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. Finally, the *BA* select the *SA*(s) it want to trade with and starts the negotiation process. The *MA* in both sides (broker and seller) can additionally enforce some trading rules based on the participant's reputations, past experiences and local allocation policies, filtering negotiation requests and responses.

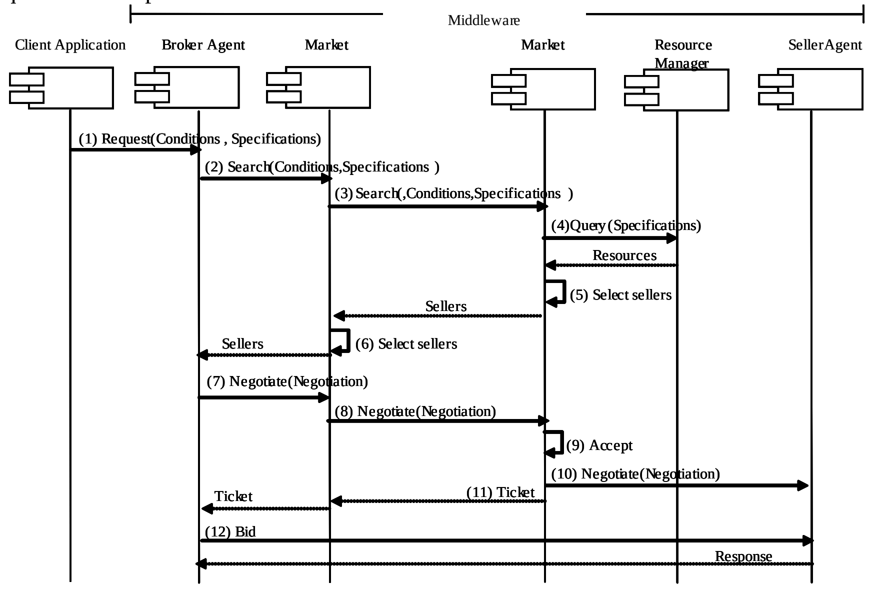
Middleware
Client Application
Broker Agent
Market
Market
Resource Manager
SellerAgent
(1) Request(Conditions , Specifications)
(2) Search(Conditions,Specifications)
(3) Search(,Conditions,Specifications)
(4)Query (Specifications)
Resources
(5) Select sellers
Sellers
Sellers
(6) Select sellers
(7) Negotiate(Negotiation)
(8) Negotiate(Negotiation)
(9) Accept
(10) Negotiate(Negotiation)
(11) Ticket
Ticket
(12) Bid
Response

Figure 5. Negotiating for resources.

3.4. Ongoing prototype implementation

Out of the layers of the architecture, the P2P Agent Layer is currently being implemented. This prototype can be viewed as an early validation of the proposed architecture with a threefold objective. First, test to what extent the middleware can be constructed using already existent toolkits. Second, validate the feasibility to compose the middleware following the proposed separation of concerns in multiple interacting agents. Finally, allow to test that the 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 implementation of the middleware builds on the use of different middleware toolkits, namely the DIET agent platform [Diet05], JXTA [Jxta05] and the WSRF/OGSA implementation of Globus Toolkit 4 [Glob05]. DIET provides a modular, lightweight and scalable execution platform, JXTA offers a rich P2P networking environment and GT4 provides full support for resource management in different scenarios. A detailed description of the selection of middleware toolkit is given in [Catn05].

4. Related work

Many market based resource allocation systems have been proposed in the literature [YeBu04]. However, all of them fail to entirely fulfill two key features needed in a resource allocation mechanism for autonomic systems: fully decentralization and openness to evolutionary environments.

The vast majority is based on a sort of bidding or utility maximization process and relay in a facilitator to accomplish the allocation of resources, introducing a high degree of centralization. One example of this approach is the GridBus project [BuVe04], which applies concepts from the utility markets (e.g. power market) for resource allocation in grid applications. GridBus is based on a Service Market Directory, where application services are published, and a Service Broker, which matches the requests from users to the available resources considering the execution const and diverse QoS parameters and looking for the optimization of the system wide utility. Our model, on the contrary, is a fully decentralized direct bargaining between producers and consumers and does not require any centralized market mechanism. This decentralization brings a higher scalability and a better adaptability to local resource requirements and to highly dynamic environments. The drawback is, however a less than optimal allocation of resources [Catn03].

Some few decentralized frameworks have being proposed in the literature, remarkably OCEAN [PHP+03] and Tycoon [LHF04]. OCEAN (Open Computation Exchange and Network) provides an open and portable software infrastructure to automated commercial buying and selling of computing resources over the Internet. Each OCEAN node that wants to buy resources uses a Matching service, which implements an optimized P2P search protocol, to find a set of potential sellers based on the description of the resources being requested. Then, an automatic negotiation process starts with each seller, based on the rules dynamically defined in a XML format. The ability to define negotiation rules is a remarkable characteristic of OCEAN that allows the adaptation of the economic model to diverse applications. The main limitation we found in this rule based approach is the lack of mechanisms for learning and adaptation to evolving environments. We found an agent based approach more suitable to achieve this level of adaptativeness.

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 agent different applications. Tycoon however doesn’t offer any framework for the construction of those agents.

A major limitation of Tycoon is that the resource allocation mechanism is already fixed in the system design and no extension or adaptation methods are offered. To overcome this limitation, our proposed framework is capable to plug key components to adapt to specific application domain in environments with heterogeneous or changing resource allocation requirements. Also, we offer a set of high level tools to develop those components, alleviating the implementation burden for new market designs.

5. Concluding remarks

We expect that the proposed architecture could guide the implementation of future large scale distributed applications which integrate decentralized and autonomic resource allocation components, employing economic mechanisms.

The proposed architecture brings a set of important benefits for the implementation task, namely an appropriated separation of concerns that will facilitate the implementation process, a great deal of flexibility and a strong “agnosticism” regarding the underlying platforms, application domain and economic model, which will make more adaptable to evolving environments.

However, we believe that some critical issues that must still be addressed, which constitutes our proposed research agenda in the field:

- A flexible framework that allows a consistent view and management of resources using a uniform set of abstractions, independently of the how each base platform handles the allocation and monitoring of its resources.
- A generic interface to pass the description of the resource requirements along with the desired conditions (preferences) from application layer to the economic agents and to automatically fill any missing information that can not be provided by the application could be automatically filled. One example of such information is the application’s budget to negotiate for resources. This brings some important consideration for the mapping from generic economic parameters (e.g. price) and the underlying technical parameters in the base platform (e.g. CPU workload).
- A set of interaction patters between the P2P Agent Layer and the Economic Algorithms Layer, to allow the adaptation of the trading network and search mechanisms to the results of the economic negotiations and the system’s performance.
- Implementation of a fully decentralized accounting and payment service to handle the user budgets and execution costs, to incentive cooperation and prevent the “free riding” of the system.
- Definition of metrics to measure the performance of the system and model to analyze them from both a technical and economic perspectives and their instrumentation in the middleware.

6. References

[ACCA04] Autonomic Communication, Report on FET consultation meeting on Communication paradigms for 2020, Brussels, 3-4 March 2004. http://www.autonomic-communication.org/publications/doc/AC_report-mar04.pdf

[BuVe04] R. Buyya, S. Venugopal (2004), "The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report, Technical Report, GRIDS-TR-2004-2", University of Melbourne, Australia, April 2004

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

[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. http://www.catnets.org

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

[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

[EyPa00] T. Eymann and B. Padovan. "The Catallaxy as a new Paradigm for the Design of Information Systems". Proc. of The World Computer Congress 2000 of the International Federation for Information Processing. 2000.

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

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

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

[LHF04] K. Lai, B. A. Huberman, and L. Fine, "Tycoon: A Distributed Market-based Resource Allocation System," HP Lab, Palo Alto, Technical Report cs.DC/0404013, Apr. 2004.

[PHP+03] P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. P Frank and C. 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

[WHW+04] S. White, J. Hanson, I. Whalley, D. Chess, J. Kephart (2004),"An Architectural Approach to Autonomic Computing", International Conference on Autonomic Computing

[YeBu04] C. S. Yeo and R. Buyya, A taxonomy of market-based resource management systems for utility-driven cluster computing, Technical Report, GRIDS-TR-2004-12, University of Melbourne, Australia, December 8, 2004.