A Hayekian Self-Organization Approach to Service Allocation in Computing Systems

Torsten Eymann *(corresponding author)*, Michael Reinicke

Chair of Information Systems (BWL VII), University of Bayreuth, 95440 Bayreuth, Tel. +49-921-55-2807, Fax +49-921-55-2216.

{torsten.eymann, michael.reinicke}@uni-bayreuth.de

Felix Freitag, Leandro Navarro, Oscar Ardaiz, Pau Artigas

Computer Architecture Department, Polytechnic University of Catalonia, Spain

{oardaiz, partigas, felix, leandro}@ac.upc.es

Abstract

Future "on-demand" computing systems, often depicted as potentially large scale and complex Service-Oriented Architectures, will need innovative management approaches for controlling and matching services demand and supply. Centralized optimization approaches reach their bounds with increasing network size and number of nodes. The search for decentralized approaches has led to build on self-organization concepts like Autonomic Computing, which draw their inspiration from Biology. This article shows how an alternative self-organization concept from Economics, the Catallaxy concept of F.A. von Hayek, can be realized for allocating service supply and demand in a distributed "on-demand" web services network. Its implementation using a network simulator allows evaluating the approach against a centralized resource broker, by dynamically varying connection reliability and node density in the network. Exhibiting Autonomic Computing properties, the Catallaxy realization outperforms a centralized broker in highly dynamic environments.

Keywords

Autonomic Computing, Digital Business Agents, Agent-Based Computational Economics, Catallaxy, Service-Oriented Architecture, Resource Brokering

1. Introduction

The focus of this article is on the presentation and evaluation of a self-organization mechanism for allocating resources in a Grid-like Application Layer Network (ALN). ALNs are software architectures that allow the provisioning of services requiring larger amounts of resources, which can be obtained from computing systems connected over simple communication infrastructures such as the Internet. In general, Grid computing, Peer-to-Peer networks, On-Demand Computing and Service-oriented Architectures can be subsumed under this category. A particular resource allocation problem in these concepts is how to match the distributed demand for a service, with an existing, but unclear supply situation.

Using self-organization for such computing system problems, instead of a centralized matchmaker, has recently gained attention by the start of large industrial research concepts like IBM's *Autonomic Computing* or HP's *Adaptive Computing* initiatives. The key motivation aspect for self-organization lies in the increasing size and complexity of today's information systems, which has led to a non-negligible growth of their control costs. Autonomic Computing uses a biological paradigm as a design and control metaphor, the autonomic nervous system {Kephart #111}. The core properties of the Autonomic Computing concept, the CHOP circle of self-configuring, self-healing, self-organization and self-protection is an electronic realization of the respective mechanisms of the human body.

Abundant biological paradigms distract from the existence of self-organizing resource allocation mechanisms elsewhere, which could, and have been used for engineering and controlling computer systems. In the physical world, for example, the proven ability of a free-

market economy to adjudicate and satisfy the conflicting needs of millions of human agents recommends it as a decentralized organizational principle {Eymann, 2004 #95; Kephart #1094; Wellman #1715}.

Applying Economic concepts to allocating or scheduling resources in computing systems is not a new idea (see {Huberman, 1988 #1012; Clearwater #625} for overviews). An early attempt at using economic ideas have been Agoric Open Systems (AOS) {Lavoie #118; Miller #1241}. AOS were defined as software systems that use market mechanisms for resource allocation, and encapsulate information, access paths and resources in objects traded by economic actor processes. Similar projects have been Mariposa {Stonebraker #1807}, Popcorn {Regev #1808}, and Spawn {Waldspurger #1809}.

The basic problem can be characterized by having a number of processors, supplying computing power to a demand situation composed of computation jobs. The particular question is how supply and demand can be matched to each other, if the actual situation on both sides is unclear. In closed environments, e.g. parallel computing, this question usually can be assumed away, as the number of processors is fixed and the arrival of computational jobs is deterministic.

However, the advent of large, open distributed networks of processors, like in Grid computing, has spurred new interest in this question. Generalizing, to match a particular computation request to a processor service in a Grid, four phases have to be conducted: service discovery, matching requests to services, scheduling the matched services and finally execution {Krauter, 2001 #1}. Existing sophisticated approaches for service discovery have been realized using flooding algorithms or distributed hash tables (DHT) {Ratnasamy, 2001 #129; Balakrishnan, 2003 #2}. The result of the service discovery phase is a list of candi-

date service provider instances. In this article, we assume that more than one service can provide access rights, and more than one client demands access - otherwise, the matching phase would be trivial.

In the matching phase, either the client (decentralized case) or a resource broker (centralized case) have to select a match out of several possible pairings, which satisfies both parties. A typical implementation of *service selection*, out of a list of discovered candidates, is a centralized matchmaker or resource broker {Chandra #601; Foster #827; Rabinovich #1406}. The matchmaker instance selects the apparently optimal match from the list, and the requesting client receives only the resulting name. Clients and service providers update the centralized resource broker in a continuous frequency about their requests and effective availability. Satisfaction can be ideally measured either by technical parameters (fast execution time, low bandwidth usage, minimal communication overhead) or by translating these to economic metrics, e.g. utility as minimal direct access costs or as a function of waiting time saved. In principle, existing service matching mechanisms can thus be visualized as a 2x2 matrix shown in Figure 1.

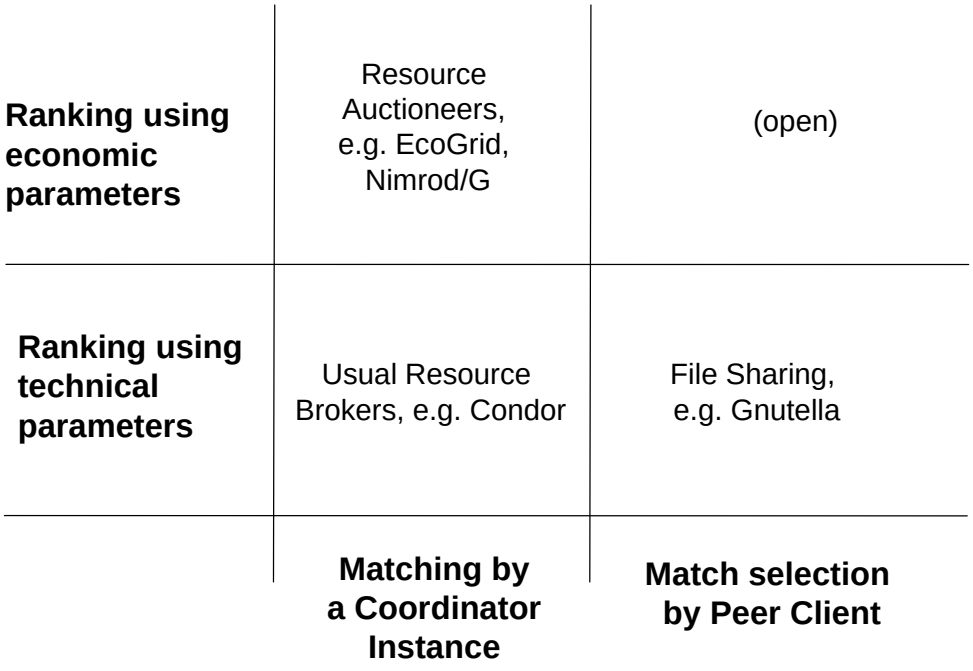

Figure 1: A portfolio of Grid Service matching mechanisms

Condor-G {Frey, 2002 #303}, Darwin {Chandra, 2001 #601}, and most Globus-based implementations {Foster, 1999 #827} typically use a centralized matchmaker instance to evaluate the candidate list. The requesting client receives one matching partner, resulting from global optimization on latency, distance or bandwidth usage, according to the current network state. Extended central approaches implement auctioneers, like in EcoGrid or Nimrod/G {Buyya, 2002 #292; Buyya, 2002 #1813}, or electronic marketplace instances {Gomoluch, 2003 #1814}, which collect bids and offers from the Grid nodes, and match supply and demand like a stock market mechanism does. Decentralized mechanisms, like in most file sharing networks, e.g. Gnutella {Adar, 2000 #383} or Kazaa, have no central point to collect supply and demand before matching. Each client decides for himself which service provider to match to based on technical parameters like estimated download time. The up-

per right corner of Figure 1, requiring client-based economic decision-making mechanisms with a model-based prediction of the system state {Gomoluch, 2003 #1814} and allocation via bargaining models {Buyya, 2002 #1813}, is only sparsely populated (one failed attempt has been MojoNation {Mojo Nation, 2003 #121}.

In the next two sections, we present an implementation of such a decentralized market mechanism concept, Hayek's Catallaxy, using a multiagent system. A comparative simulation of a Catallaxy resource allocation approach vs. a centralized approach in an application layer network indicates the strengths and weaknesses of the implementation. After that, we discuss related work on using Economic concepts for controlling computer systems, after which the article ends with the issue, whether the Catallaxy concept may be a fruitful alternative for engineering Autonomic Computing systems.

3. The Catallaxy: a self-organization concept from Economics

Friedrich August von Hayek {Hayek #952} understood the market as a decentralized coordination mechanism, as opposed to a centralized command economy. Apart from political macroeconomic thoughts, his work also provides concrete insight on the working mechanisms of economic coordination. The emergence of software agent technology and increasing size of information systems leads to the possibility of implementing Hayek's Catallaxy concept and using the ensuing "spontaneous order" as a concrete proposal for both the design and coordination of information systems. However, a formal description of this self-organizing market mechanism does not so far exist.

The Catallaxy concept bases on the explicit assumption of self-interested actions of the participants, who try to maximize their own utility and choose their actions under incomplete information and bounded rationality {Simon #1539}. The term Catallaxy comes from

the Greek word “katallatein”, which means, “to barter” and at the same time, “to join a community.” 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 {Hayek, 1989 #952}. The main characteristics of the Catallaxy {Hoppmann #1006} are that

1. Participants work in their own interest to gain income. Every system element is a utility maximizing entity, which requires the definition of utility itself, of means to measure and compare income and utility, and to express a desire to reach a defined goal. For humans, these definitions have not necessarily to be explicit or thoroughly defined; for information system elements, this explicitness is required.

2. Participants subjectively weigh and choose preferred alternatives in order to reach an income or utility maximization goal. In economic theory, the “homo oeconomicus” is a completely rational utility maximizer. He can choose an alternative action out of total knowledge about the environment. Hayek’s claim was that such an “objective” choice is not possible because of "constitutional ignorance”, that it is (inevitably) impossible to know each and every detail of the environment state. For large and very dynamic information systems, this is inherently true, and overcoming it by central means requires synchronization and restriction of possible actions of the single elements.

3. Participants communicate using commonly accessible markets, where they barter about access to resources held by other participants. The development of prices for a specific good, whether they are increasing or decreasing, leads buyers to look for al-

ternative sources of procurement and thus enhances the dynamics of the market. Note that a market here is nothing more than a communication bus – it is not a central entity of its own, which collects all information and matches market participants using some optimization mechanisms, which would contradict "constitutional ignorance".

In human economic systems, these institutions are implicit; for a realization in distributed information systems, the properties of utility maximization, strategies and the exchange of offers need to be explicitly specified and implemented. Formal descriptions for using economic mechanisms in distributed computing systems can be found in {Ferguson, 1996 #796}.

As a blueprint for other possible forms of Service Grids {Gomoluch, 2003 #1814} or Application Layer Networks, we describe the concept using a simple web services scenario of a PDF conversion service {Catnet Project #85}{T-Online AG #137}:

Adobe's PDF file format is a common exchange file type for mixed text and graphics documents, mostly due to its preservation of layout specifics. The files are created using the (usually locally installed) Acrobat Distiller service, which converts from e.g. Microsoft Word or Postscript files. In an "on-demand" Service Grid, Distiller web services are available in the network, hosted by independent vendors and directly accessible from the software application, competing with each other for the clients' demand. The word-processor client programs transparently address such a networked PDF conversion service instance in the background, without disturbing the user's course of work. Clients and service provider instances bargain on access prices on a case-by-case basis, taking into account the current and prospective development of supply and demand to increase the mone-

tary utility of their respective owners. Services instances are situated on host computers, which, for simplicity, are assumed to provide processor power and storage on a fixed cost basis.

Economic actors are straightforwardly implemented as intelligent software agents {Wooldridge #1759}. Agents are embedded in an *environment*; whose state they experience through *sensors*; which lead to a comparison of an actual environment state with a desired environment state using an *internal world model*; and where they try to influence the environment state using *effectors* towards that more desirable state.

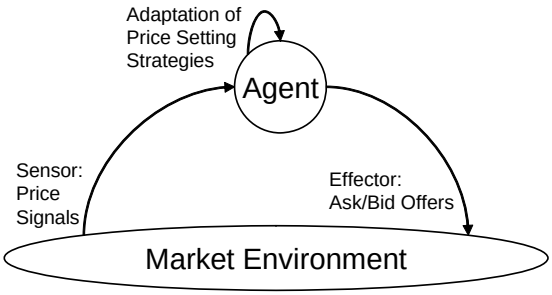

Figure 2: Properties of Digital Business Agents (cf. {Wooldridge #1759})

Figure 2 shows a *Digital Business Agent* {Eymann #301} working in a market environment. Sensors and effectors are realized as price signals incoming from and outgoing to the market environment. If the agents' utility goals are not met by the present ownership situation, they negotiate with each other in order to maximize utility by exchanging resource access rights (e.g. using an alternating offers protocol {Rosenschein #1445}). Bartering forms a sequence of effectors and sensors, this leads under partial and bounded knowledge to an adaptation of the agent's internal model. Implementing Edgeworth bartering {Varian #1674}, the agents trade bilaterally and secretly with each other, *if* the internal world model

prognoses utility increase out of the potential transaction. Setting the price right is the most important action decision. Sellers intent to obtain the highest possible price for the service access they offer, buyers want to pay the lowest possible price for the service. To that respect, the seller offers (*Asks*) will be higher than the reservation prices, while buyer offers (*Bids*) will probably be lower. A too high price in the face of competition will not lead to many transactions, while a price too low leads to less income per transaction.

The constant price signaling between entities propagates changes in the scarcity of resources throughout the system, and leads to constant adaptation of the system as a whole. Imperfect knowledge makes it thus necessary to adapt the agent's price setting strategies dynamically in order to maximize individual utility. This is achievable using feedback learning algorithms (Evolutionary algorithms, Numerical optimization, or hybrid methods like Brenner's VID model {Brenner #539}, which are all principally interchangeable {Müller #125}).

In our example, three types of market agents appear (see Figure 3): the client agents, the service instance agents and the resource agents (as embodiments of the hardware/network provider). The market environment itself is not a solid object – it is a communication platform, implicitly realized by the network provider, communicating the effector actions of all other agents in the environment.

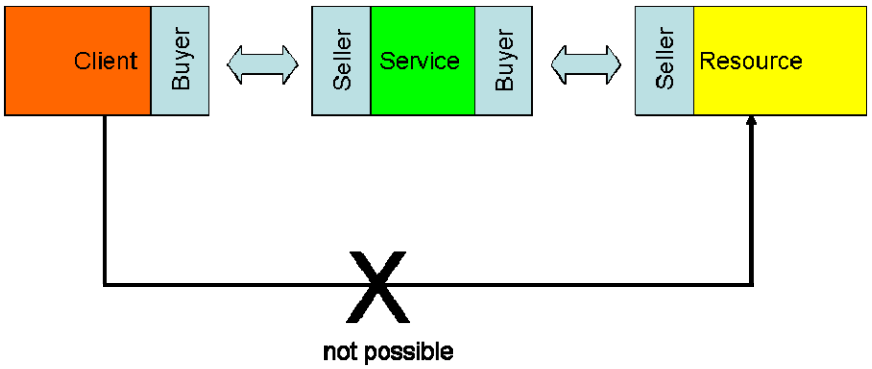

Figure 3: Interaction relations in the Service Market

- The *Client* is a computer program on a certain host, which needs access to a web service to fulfil its design objectives. It tries to access that service at an arbitrary location within the computer network, use it for a defined period, and then continues with its own program sequence. Client programs run on a connected network resource, usually a personal computer. The business strategy of the client computes the fixed cost of purchasing a local copy of Adobe Distiller against the variable cost of using such available On-Demand services, multiplying the forecast number of uses with the access price. If the client user does not need to convert to PDF so often, it will rely on a certain price and availability level of the service provider and refrain from buying an expensive local copy; if the usage frequency is above a certain number, it is in total cheaper to buy that exclusive copy. In an environment where services have to be paid for access, the utility gain of clients is the difference between their *private value* (of what the access is worth) and the actually paid *transaction price*:

$$u_i^C = v_{i,p} - p \tag{1}$$

- A *Service* is an instantiation of a general application function, embodied in a computer program. A *Service Copy* is an instance of the service; a resource computer, which provides both storage space and bandwidth for the access of the service, hosts it. The busi-

ness model of the service provider leads to set the access price so that the majority of clients decide to rely on the on-demand option. In theory, the price will be equal to the marginal cost of processing the penultimate access, which means that it is exactly so high that the consumer is undecided whether to buy the local copy or access the remote service (provided that both prognoses the same number of accesses in a given time span). If the service provider is able to distribute several instances (service copies) in the network, he might be able to sell each copy access for a different price, according to the time of day, the geography of the network or the willingness of clients to pay. Each redundant web Service Copy is thus a miniature business, like a retailer's branch store. Like the clients, the service providers also have a private value for service access (*sa*) . In addition, there is private value for buying network resource access (*ra*) from the hosting node:

$$u_j^{SC} = \left(p^{sa} - v_{p,j}^{sa} \right) + \left(v_{p,j}^{ra} - p^{ra} \right) \tag{2}$$

- A *Resource* denotes a host computer, which provides a limited number of storage space and access bandwidth for service transmission. The network connections between the resources are simulated to be of equal length and thus of equal transmission time and costs. The resources and network owner (the network provider) allows service providers and clients to communicate using cables, routers, gateways and other, hardware or software network layer instances. For the usage of these resources, he gains income from all participants – the more participants, the more money can the network provider make. However, more participants means more traffic in the network, and above some level the traffic can get so extensive that the existing resources are no longer sufficient. However, if the dimensioning of resources is too large, the income from the participants

might not be high enough that the resource investment is economically justified. In the long run, the network provider will provide enough network resources for the average use, but will be vulnerable to usage spikes. These resources incur costs, and the network provider aims to fill these costs and to make profits by increasing the usage of the resources:

$$u_k^R = p \ - v_{p,k} \tag{3}$$

Summing up all utility functions over the number of respective participants, the parameter Social welfare utility (SWF) measures how the aggregate of all the individual utility is maximized. The equation thus can be written as

$$U_{SWF} = \sum u_i^C + \sum u_j^{SC} + \sum u_k^R \tag{4}$$

After each successful trade, the sum of all utilities of all participants increases. A fictive final state would have maximum overall utility and is Pareto-optimal, which means that no single agent can propose a change that does not decrease any other's utility. However, as ALN nodes appear and disappear dynamically, such a solid state may never be reached. Under the restriction of an imperfect knowledge situation, a total optimal value of SWF can only be measured in hindsight.

5. Simulation of a Catallactic application layer network

In this section, we compare two methods of allocating resources using a network simulation, while varying the number of active network nodes and the dynamics of their connection to the network. The purpose is to evaluate how a centralized resource allocation method, which depends on total knowledge, performs in comparison to a decentralized resource allocation method working on imperfect knowledge, in large and highly dynamic

environments {Catnet Project #85}. Our CATNET ALN simulator allows comparing two main resource allocation strategies: a “baseline” control mechanism and a “Catallactic” control mechanism.

The baseline control mechanism computes the resource allocation decision employing a centralized resource broker instance, using a sealed-bid double auction in continuous time intervals. Location and interfaces of the dedicated service coordinator (the master service copy, MSC) are known to the individual service copies (SC). Whenever a client broadcasts a *request* in the network via the connected resource hosts, the MSC gets the request forwarded. Out of its knowledge of network status, the MSC is able to compute the costs of providing a service, ranks those virtual offers of all SCs, and sends back an *accept* message revealing the cheapest SC to the client. Settling the payment for the service invocation would be a matter of the individual SC and the client in an “on-demand” economy.

In the Catallactic mechanism, no MSC exists. The clients broadcast their *requests* over the network in a Gnutella-like fashion, trying to reach as much service copies as possible, within the diameter of the message’s hop counter. Any available SC, which receives the request message, returns a *propose* message containing the initial price for provisioning the service. The client orders all incoming proposals in its inbox and, beginning with the best offer, engages in a bilateral alternating offers protocol until acceptance or final rejection, in which cases he continues further down the ranked list of offers.

The key ingredient to achieving Catallaxy lays in the computation of the initial offer prices of the seller and buyer agents. As reception of all requests cannot be guaranteed, the SC agents have only partial knowledge of the market status. They thus can not exactly compute the true supply function. A self-interested strategy tries to estimate the supply function,

adds some slack for uncertainty, and some percentage for one's own utility maximization. Inevitably, the actual demand and supply curves will lead to a small, inevitable market efficiency loss.

However, the feedback from the market leads to continuous adaptation. The *effect* of issuing some offer into the market is followed by *sensing*, whether that offer has lead to a successful transaction or not (cf. Figure 2). If the offer of the SC is turned down, the price may have been too high – the adaptation of the negotiation strategy leads to relaxing the initial price. On the other hand, if the SC's offer is successful, the initial price can be raised to gain more income from the transaction. The agent's strategy thus follows the dynamics of supply and demand, as indicated in the following example:

In the PDF example, the Catallactic mechanism works by giving the requesting client an argument for selecting one conversion service over the other. Given conversion service instances A (demanding an access fee of € 0.12) and B (demanding € 0.14) and all other parameters being equal (such as communication cost, processing time etc.), client C with a subjective market price of € 0.13 would naturally select instance A. Both A and C realize a utility increase of € 0.01. The economic interplay of supply and demand, implemented as an adaptation procedure in the agent's strategies, now leads to emergent coordination: the self-interested service instance A tries to raise the access fee to € 0.13, because it was successful, while instance B lowers its price to € 0.13 for just the opposite reason. The self-interested client instance C adapts its subjective market price to € 0.12, preventing a steady raise of the provider prices. In future negotiations and with thousands of similar procedures in parallel, the market price will accordingly settle at a steady level, providing emergent and stable coordination to the network as a whole.

In our simulations, we have analyzed changing ALN environments by varying the network setup using the dimensions of node density and node dynamics. Node density measures the number of service copies available in the network – the more SCs can provide service access, the denser is the network. Node dynamics measures the probability that SCs can become disconnected and reconnected again – a static network shows a constant availability of 100%, while a peer-to-peer network is practically defined by continuous appearance and disappearance of nodes. The intention of choosing these dimensions was to capture different types of ALN environments in one simulation, while restricting the number of simulations that need to be run. In the current model, for each change in the underlying variables, we have run 50 simulations: 25 for Catallactic and 25 for Baseline resource allocation. As simulation input we use a trace of client demands containing service requests. Each service request specifies the amount of service, a price, and the duration of the invocation. All simulations use the same demand trace.

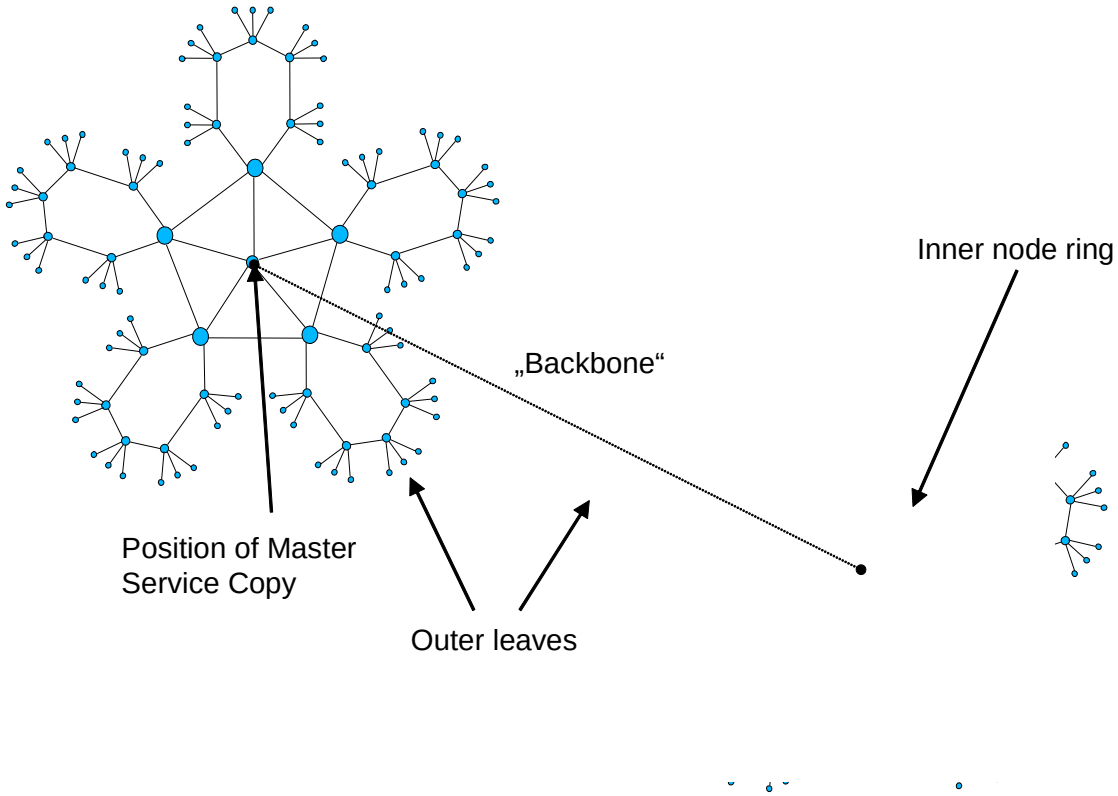

Figure 4: Network topology

The physical network topology used in the simulations is organized in two rings, each having three levels of pentagons with leaves on the outer level (Figure 4). The rings are connected at the inner pentagon nodes; the intention of that topology is to resemble current ALNs, which mainly consist of two networks in North America and Europe, connected by a few backbones. The ALN is built on top of the physical network. Network nodes are instantiated having one of the previously described types, being a client, service copy or resource agent. Depending on the particular simulation, a node may contain several agents or no agent at all. In the second case, the node acts as a router. Table 1 shows the configuration of the simulations.

Table 1: Description of the Simulations

Input trace	- 2000 service requests generated randomly by 150 clients over a time interval of 100 s. - time between issuing requests is 75 ms.

	- each request is for 1 service unit. - each request message can travel 10 hops distance maximum. - each service invocation has a duration of 5s, in which the SC is blocked. - Cycle time of the master service copy, if active, is 150ms.
Topology	- 212 physical nodes in 2 rings.
Node density	Always 150 clients on the leaves of one physical network ring. Different density applies to resource and service copy agents. - Density 0: 6 nodes hold 50 service copies each. - Density 1: 12 nodes hold 25 service copies each. - Density 2: 25 nodes hold 12 service copies each. - Density 3: 50 nodes hold 6 service copies each. - Density 4: 75 nodes hold 4 service copies each.
Node dynamics	Dynamic behavior: On start, 70% of the service copies are connected. - Dynamics 0 : Service copies do not change their state (static network) - Dynamics 1: Every 200 ms any service copy can change its state (connected/disconnected) with a probability of 0.2. - Dynamics 2: as before with a probability of 0.4 - Dynamics 3: as before with a probability of 0.6 - Dynamics 4: as before with a probability of 0.8

The following results are from the same simulation. The first question is how the system behaves with respect to the Resource Allocation Efficiency (RAE), which measures the percentage of successful negotiations. In an Edgeworth barter setting, every successful negotiation indicates an increase in overall utility. The metric thus allows a statement on the overall performance of the system. The total values of the scale are artifacts of the simulation setup, but the relative performance of Catallaxy vs. Baseline is repeatable.

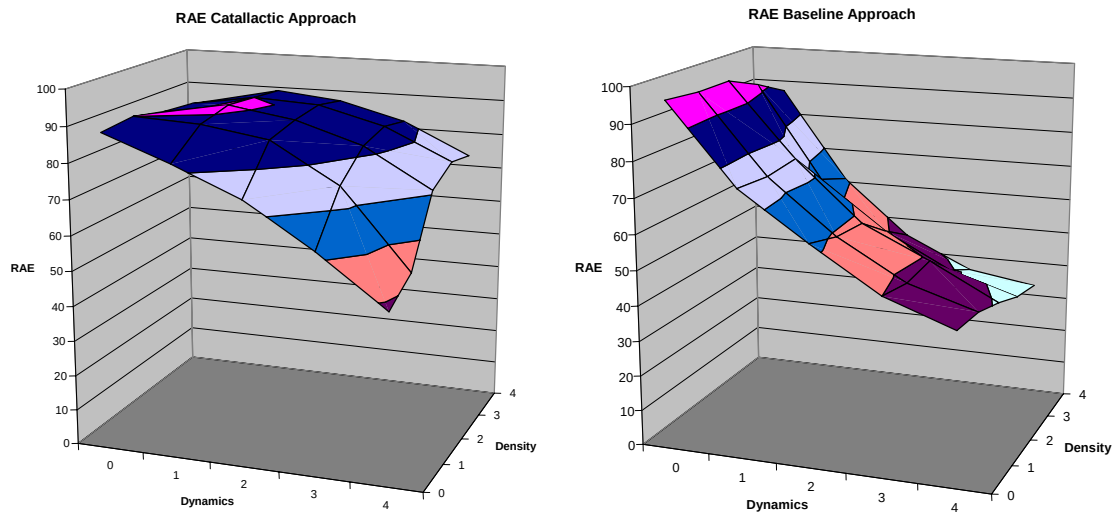

Figure 5: Resource Allocation Efficiency for Catallactic vs. Baseline Simulations

The left side of Figure 5 shows the results of the decentralized, Catallactic resource allocation approach. The two axes at the bottom reference to the changes in node density and network dynamics, as explained in Table 1. The RAE performance decreases with increasing dynamics, the worst RAE occurs when density is low (only few nodes exist) and dynamics are high. In comparison with the Baseline approach shown on the right side, the slow decrease in overall performance is notable. For a static network with low dynamics and low density, both mechanisms fare quite well, as there is ample time to discover all available services, negotiate with them and decide on allocation. Changes in the density of the resources have only small effect on the Baseline RAE. As the central Baseline broker does the main work, the geographical layout of the network seems to play only a minor role. With increasing dynamics, however, the Catallactic RAE is less affected.

If we look closer at technical parameters, which are known to be inferior for distributed systems and bargaining approaches, we find a different picture. An example is the response time (REST), which marks the time span between issuing a demand request and the final satisfaction or rejection of that request (timed-out negotiation attempts in-between add up to the total). Figure 6 shows average response times for the successful and accepted service negotiations, with the Catallactic approach to the left. When comparing Catallaxy here to Figure 5, the outcome is quite diverse: Except for high dynamics in a low density regime, the outcome seems to be relatively stable. A REST of approx. 600ms is the maximum, which is probable an artefact of the flooding algorithm, searching to get an adequate number of replies. At higher density levels, and seemingly independent of the dynamics level, we find a minimum below 320ms. The decentralized approach would probably benefit from using a revised discovery algorithm like CAN or Chord in low density and dynamics regimes.

The right side of Figure 6 shows the response time of the Baseline mechanism, resulting in a rugged landscape. The outcome is determined by the cycle time of the master service copy (MSC), which synchronizes the existing market situation and concentrates it at one location. Dynamics thus has an ample impact on the response time, with a twofold increase over the simulation range. This is probably caused by higher dynamics leading to more misallocations and therefore to time-consuming new requests and allocations. Density variations have only a slight effect on the response time, because the geographical distances in the network are equalized by the MSC.

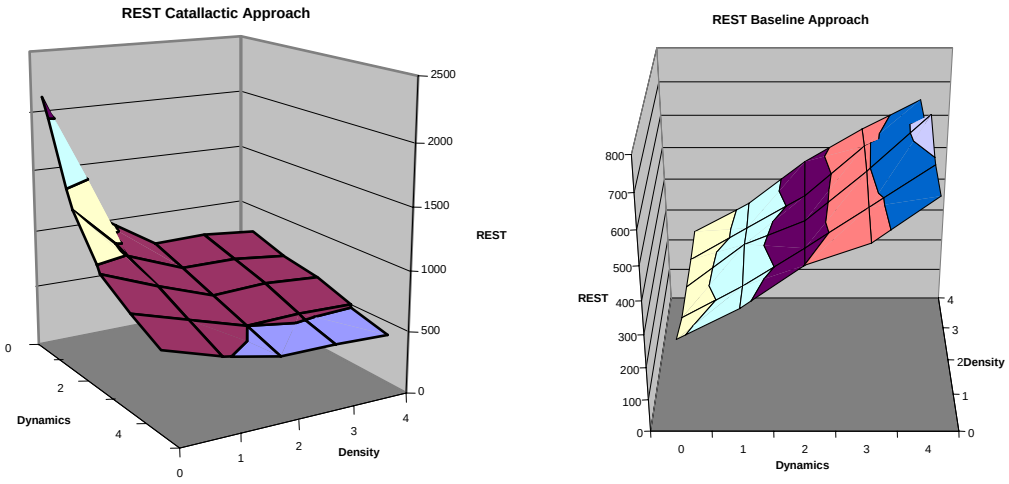

Figure 6: Response Time for Catallactic vs. Baseline Simulations

6. On implementing Economic Self-Organization in Computing Environments

What makes Economics so attractive for computing environments is that its central research question lies in the effective allocation of resources, provided by suppliers and in demand by customers. In computing environments like Grid Computing, the resources in question are processor time or storage space, while the economic actors are computers or web services {Buyya #574; Buyya #292}. It appears that, by just implementing markets in computing environments, the satisfying ability of economics might be viable for creating cost-effective computer architectures.

However, between the mostly descriptive economic concept and the normative technical implementation lies a fundamental gap, requiring selective choice of how actors, resources, goods, and markets are modelled and embedded in a technical environment. Some researchers call this task “market engineering” {Weinhardt #144}. The basic purpose of market engineering is to capture the inherently decentralized, dynamic coordination nature of the economic concept, and to translate that into a technical realization, which allows optimizing resource allocation.

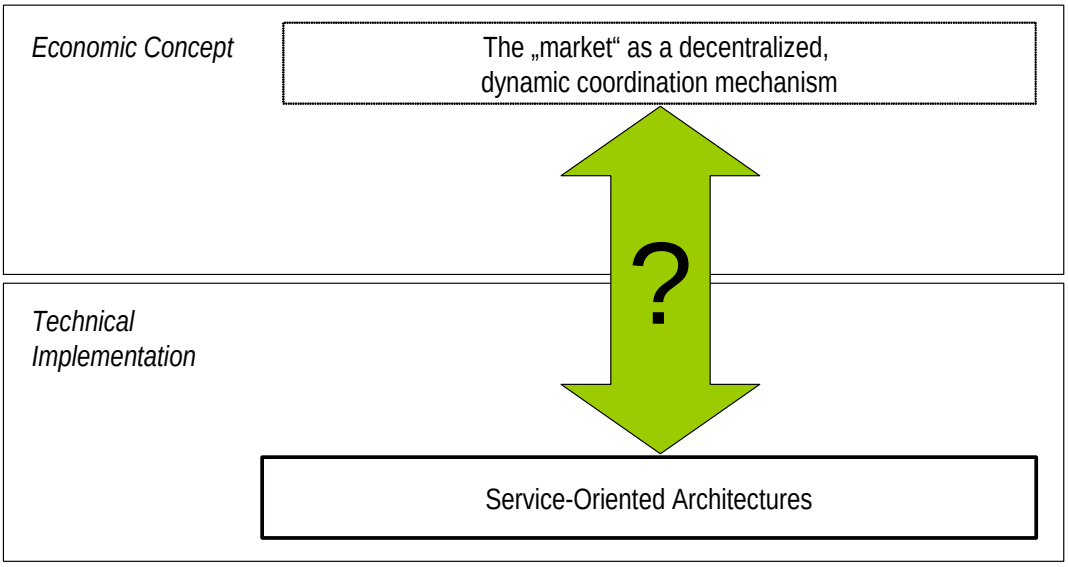

Figure 7: Realizing economic concepts in a technical implementation

There are several competing descriptive approaches to how economic resource allocation mechanisms work. In general, Economics is essentially all about the coordination of systems consisting of utility-maximizing agents, who satisfy their needs using some *mechanism* for solving a distributed resource allocation problem. The *effect* of this mechanism is a

state where prices are set, so that supply and demand is perfectly balanced, and the number of transactions is maximized {Kearney #1088}. All implementation attempts try either to recreate the mechanism, or to achieve the effect by using another mechanism, adding some side condition like zero communication costs or a steady environment state.

Adam Smith's proverbial *invisible hand* {Smith #1551} was a first concept of a decentralised mechanism without a co-ordinator, but Smith gave no implementation of that mechanism. A century later, Leon Walras {Walras #1690} introduced a central auctioneer, who iteratively solved the allocation problem out of total knowledge of supply and demand. With this mechanism, Walras was able to generate the desired equilibrium effect.

Most of today's economic research relies on Walras' *tatônnement* process as a valid picture of the mechanism, which influences also the possible realization in computing environments. An example is the realization by Wellman {Wellman #1715}, titled Market-Oriented Programming (MOP), in an distributed artificial intelligence (DAI) environment. Wellman takes the notion that „an economy is a multiagent system" literally; the distributed agents individually compute their utility functions and post that information to a centralized Walrasian auctioneer. During the computation process, interrelated markets are successively brought to near-equilibrium by the auctioneer, with the final general equilibrium effect as the „gold standard" to achieve. MOP has been successfully used in electricity markets {Ygge #1771}, for multi-commodity flow problems {Wellman #1714}, supply chain management {Wellman #145} or for negotiations about the quality of service in multimedia networks {Yamaki #1768}.

In contrast, Economics research on self-organization still aims at explaining the mechanism of the invisible hand, e.g. Agent-based Computational Economics {Tesfatsion #1611}. Ac-

tually, there is growing interest in using self-organization, as indicated by the start of large industrial research concepts like IBM's *Autonomic Computing* initiative. Autonomic Computing uses a biological paradigm as a design and control metaphor, the autonomic nervous system {Kephart #111}. If the mechanisms underlying Hayek's *spontaneous order* concept {Hayek #952} can be properly understood, it might be possible to build large Autonomic information systems using the Catallaxy approach, where artificial entities coordinate themselves, just as human economy participants do in the real world. For a start, we have to discuss whether the desired effects of Autonomic Computing are achievable (and describable) using economic terms.

IBM's Autonomic Computing Manifesto {IBM #312} describes seven characteristics, which self-adapting systems should exhibit. The core characteristics are contained in the so-called CHOP cycle of self-configuring, self-healing, self-optimizing and self-protection capabilities. The *self-configuration* property is indicated in the variation of prices when adding or removing service providers (cf. the different density regimes). The *self-healing* of the system is apparent in case a service provider instance shuts down or a network connection gets broken (cf. the different dynamics regimes). The application is *self-optimizing*, in that the agents constantly attempt to change their strategies towards the maximum utility-eliciting negotiation positions, which respectively lay on the total supply and demand curves. The *self-protection* of the application finally can be reached by including security mechanisms like reputation tracking {Eymann #300}, which are effective in separating malicious and underperforming agents.

In addition, viewing Autonomic Computing systems as Economic systems has some merits, too. The main applications for AC systems will be deeply rooted in a business context. With a biological background, you need to find biological translations for conceptual data

structures and functionality for describing success, utility, or business goals. This is not a trivial process, and may lead to semantic loss underway. For example, the business goal of maintaining availability (to prevent loss of profit in the case of server downtime), may be translated biologically as “staying alive”. However, the semantics of both differ – deliberately shutting down a biological AC system may qualify for murder, while in economic terms, shutting down a system means buying it out of business – with the programmer defining what the currency is.

7. Conclusion

The CATNET simulation presented in this article is a generalizable network simulation for application layer networks. We have used its specification and the properties of the resource allocation mechanism to investigate into constructing Autonomic Computing systems using an Economics, rather than a Biology approach. Our simulation results indicate that resource allocation in networks, where many small nodes work in a highly dynamic environment, can be coordinated successfully by the Catallaxy paradigm. The results presented in this article are only the first step towards successfully engineering “economic” autonomic computing systems (see also {Cheliotis #1810}). On the engineering side, we found optimization potential in the design of the negotiation protocol or in the possibility to relocate underperforming service copy instances to other hosts.

The key to this semantic shift is to view the „market“ as an emergent mechanism of coordinating and matching supply and demand offers, and market participants as rationally-bounded, self-interested individuals, like in Neo-Austrian {Hayek #952} or Neo-Institutional Economics {North #1308}{Furubotn #1811}. As we move on to technology, which allows us to map unstructured semantic knowledge in large and very dynamic sys-

tems, we have to look for decentralized self-organization, not at least for reasons of exponentially increasing „costs of ownership" {Truex #1639}.

Given a highly complex and dynamic ALN infrastructure, scalability and the management of a great number of heterogeneous resources are supposed to be challenging issues of future ALNs and computing systems in general. Ubiquitous computing {Weiser #1712} envisions trillions of computing devices connecting and interacting with each other; Grid Computing {Snelling #131} envisions millions of networked processors. To handle the complexity and scale of such systems, the necessity of a centralized management could easily turn the vision into a "nightmare" {Kephart #111}. The solution is not necessarily a question of overcoming semantic gaps or problems of multi-attribute optimization. Discovering and selecting web services from huge numbers of unreliable candidates alone is challenging enough.

7. Acknowledgements

The CATNET project has been funded by the European Union under contract IST-2001-34030. We thank two anonymous reviewers for valuable comments.

Appendix

Vitae

Torsten Eymann holds the chair of Information Systems at the University of Bayreuth, Germany. He studied information systems in Kiel and Mannheim and received a PhD from Freiburg in 2000. Together with Deutsche Telekom (T-Systems), he published the technology study "Digital Business Agents" in 2000. He is author of 2 books and several journal and conference papers, some of which received best paper awards. For longer research stays, he has visited Hitachi's System Development Labs, Yokohama, Japan, and British

Telecom Labs, Martlesham Heath, UK. In 2003, he co-organized the first European workshop on “Autonomic Computing” in Prague.

Michael Reinicke received a diploma in business administration from the University of Wuerzburg in 2001. Since February 2002 he was a PhD student at the University of Freiburg, and since September 2004 at the University of Bayreuth. His research topics are economic software agents in application layer networks, economic organization of highly dynamic mobile networks and Peer-to-Peer architecture. He has co-authored several conference papers.

Leandro Navarro obtained the PhD in Telecommunication Engineering from the UPC in 1992. He is Associate Professor at the UPC teaching on distributed system architecture. His research interests are collaborative work in distributed systems, and large scale distributed systems. He has worked in a number of EU and industrial projects.

Felix Freitag received the PhD in Telecommunication Engineering from the UPC in 1998. He is full time adjunct lecturer at the Computer Architecture Department. He worked on speech recognition systems. His current research concerns the performance evaluation of distributed and parallel applications.

Oscar Ardaiz received an MS degree in Telecommunication Engineering from UPNA in 1995. He is doing his PhD at UPC with a scholarship from the Spanish Government. He made a research stay at University of South California-ISI in 1999 working on the X-Bone Project. He is currently finishing his PhD thesis on the topic of scalable service deployment.

Pau Artigas is a PhD student at UPC Barcelona in the CATNET project.

Figure captions

Figures embedded in the text for review purposes.

Tables

Tables embedded in the text for review purposes.

References