Decentralized Resource Allocation in Application Layer Networks

T. Eymann, M. Reinicke

Institute for Computer Science and Social Studies
Albert-Ludwigs-University Freiburg, Germany

eymann,reinicke@iig.uni-freiburg.de

O. Ardaiz, P. Artigas, F. Freitag, L. Navarro

Department of Computer Architecture
Technical University of Catalonia, Spain

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

Abstract

Application-layer networks (ALN) are software architectures that allow the provisioning of services requiring a huge amount of resources by connecting large numbers of individual computers. The ALN simulation project CATNET evaluates a decentralized mechanism for resource allocation in ALN, which is based on the economic paradigm of the Catallaxy, against a centralized mechanism using an arbitrator object. In both versions, software agents buy and sell network services and resources to and from each other. The economic model is based on self-interested maximization of utility and self-interested cooperation between agents. This article describes the design of money and message flows for centralized and decentralized coordination in both versions and shows preliminary results.

1. Allocation of Resources in Application Layer Networks

Application-layer networks (ALN) are software architectures that coordinate the provisioning of services requiring a huge amount of resources by connecting large numbers of individual computers. Such global Internet-based networks, like today's Grids [2] and Peer-to-Peer-Computing [14], take advantage of such infrastructures with applications like multicast services for global audiences, storage repositories of peta-scale data sets, or parallel computing applications requiring teraflops of processing power.

Such applications are executed in multiple resource locations distributed throughout the Internet, coordinated on the application layer using a dedicated network, the ALN. An ALN scenario would be the distributed provisioning of web services for Adobe's Acrobat (for creating PDF files). Here, word-processor client programs would transparently address the nearest/ cheapest Acrobat service instance in order to create PDF files. The overall objective of the ALN would be (a) to always provide access to some Acrobat service instance, such that a minimum number of service demands have to be rejected, and (b) to optimize network parameters such as provisioning and transmission costs. This paper assumes that the future development of these applications will lead to clients paying for the access to a service and the corresponding on- or offline exchange of payment; the individual goal of a client would become to access a service cheaply, while services may try to maximize income.

In order to keep an ALN operational, service control and resource allocation mechanisms are required. Their basic purpose would be to match service supply and demand, in the likely case of multiple, redundant service instances, to meet those objectives. The simple service discovery mechanisms available today in decentralized networks (e.g. Jini [19]) seldom provide such functionality, as the case of redundant service instances is yet rare.

However, a realization of these mechanisms by employing a centralized coordinator instance (auctioneer, arbitrator, dispatcher, scheduler, manager), like e.g. in GLOBUS [8] or CONDOR-G [9], has several drawbacks.

First, ALN and the underlying networks are very dynamic and fast changing systems: service demands and nodes connectivity changes are frequent, and new different services are created and composed continuously. Information collected from the network is considered to be outdated when it reaches the coordinator; any solution computed on the basis of this information tries to optimize a past and inconsistent state of the network. Dynamic ALN need a continuous, real-time coordination mechanism, which reflects the changes in the environment.

A second related property is that the coordinator should have global knowledge on the state of the network. This is mostly achieved by calculating the time steps such that actual status information from all nodes arrives safely at the coordination instance. However, if the diameter of the network grows, this approach leads to long latency times for the nodes.

Third, a centralized coordinator is part of the problem that decentralized ALN are trying to solve: As bids and offers have to route through the network to the single instance which collects global knowledge and computes the resource allocation, the distribution and deployment of services throughout the network is counteracted. This is currently not a problem as the control information is small compared to the allocation data itself, but may increase

when the principle is applied to more and more application areas.

These drawbacks lead to the search for a truly decentralized coordination concept which is able to allocate services and resources in real-time without a dedicated coordinator instance. This concept should on one hand be able to cope with technical shortcomings like varying amounts of memory and disk space, internet connection speed and sporadic appearance and disappearance of the services. On the other hand, it is desirable that the network as a whole shows optimized behavior with regard to low overhead communication, short computation times, and economical resource allocation.

Recent research in Grid computing has also recognized the value of price generation and negotiation, and in general investigates economic models for trading resources and services and the regulation of supply and demand of resources in an increasingly large-scale and complex Grid environment. A general overview on resource management and scheduling in Grids is given in [4]. However, each Grid project is differently designed, and it is not possible to compare different allocation mechanisms within the same network without changing the fundamentals.

In the remainder of this article, we first introduce a decentralized economic concept for coordination, the Catallaxy, and describe the CATNET project. The following section compares money and message flows in the application-layer network economic model, both with a centralized (baseline) and a decentralized implementation. Next we describe how the experiments are conducted in both cases. The article closes with some preliminary experimental results and an outlook to further research.

2. The Catallaxy Paradigm and the CATNET Project

The Catallaxy coordination approach [7; 10] is an economic coordination mechanism for information systems consisting of autonomous network elements, which is based on constant negotiation and price signaling. Using concepts from agent-based computational economics [18], the goal is to develop new technical possibilities of coordinating decentralized information systems consisting of autonomous software agents. The software agents are able to adapt their heuristic strategies using machine learning mechanisms [17], and this constant revision of strategies leads to a co-evolution of software agent strategies, a stabilization of prices throughout the system and self-regulating coordination patterns [6]. The resulting patterns are comparable to those witnessed in human market negotiation experiments [13].

Earlier work in the context of computer science has used economic principles for resource allocation in operating systems, packet routing in computer networks, and load balancing in distributed computer systems [5; 11]. Most of these approaches rely on using a centralized auctioneer and the explicit calculation of an equilibrium price as a valid implementation of the mechanism. A successful implementation of the Catallaxy paradigm for a distributed resource allocation mechanism promises the advantage of a more flexible structure and inherent parallel processing compared to a centralized, auctioneer-based approach. This comparison can be done using both economical and technical criteria.

For the economic evaluation of the overall success of the control mechanism we use the "maximum social welfare utility" (SWF) criterion, which is the sum of all individual utility function values of the participating nodes [16]. Every Client, Service Copy or Resource gains individual utility from buying lower or selling higher than the perceived market price. It can be enhanced by doing more transactions in the same time, but communication costs subtract from it. In total, SWF balances revenues and cost throughout the network. Increasing performance and decreasing communication in the whole network thus directly computes to relatively maximize social welfare utility. Other evaluation parameters are communication cost, allocation efficiency, network traffic and service access latency.

The goal of the CATNET[1] project is thus to evaluate the Catallaxy paradigm for decentralized operation of application layer networks in comparison to a baseline centralized system. To achieve this, we have developed the CATNET ALN simulator, which allows to experimentally compare two main resource allocation strategies: A centralized approach in which allocation decisions are taken centrally and a decentralized approach, where local agents negotiate resources using economic models.

The CATNET ALN simulator is implemented on top of the JAVASIM [3; 12] network simulator. It can be configured to simulate a specific ALN, such as a content distribution network or peer-to-peer network. Different agent types can be instantiated, namely clients, resource agents, and service agents. Network resources to be allocated encompass service access, bandwidth and storage. The simulation builds on a TCP/IP network model supported by JAVASIM. It describes the generic structure of a node (either an end host or a router) and the generic network components, which can both be used as base classes to implement protocols across various layers.

[1] CATNET is supported by European Commission Information Society Technologies Programme under contract no. IST-2001-34030.

3. Money and Message Flows in the network

During the runtime of the network, software agents in the network nodes buy and sell access to network service copies using a heuristic and adaptive negotiation strategy. Changes in prices for certain services reflect changes in the supply and demand situation, which are propagated throughout the network. Both client and service provider agents will adapt their strategies about where to buy and sell based on the received information, and thus continuously change the state of the network.

3.1 The general simulation setup

The CATNET application simulates two main control mechanisms for network coordination: a "baseline" control mechanism and a "catallactic" control mechanism. The baseline control mechanism computes the resource allocation decision in a centralized service/resource provider. The catallactic control mechanism has the characteristic that its resource allocation decisions are carried out by self-interested agents with only local information about the environment. Each agent has a resource discovery facility and a negotiation strategy module. The following class types are defined:

- Client: a computer program on a certain host, which needs access to a web service to fulfill its design objectives. The Client (C) tries to access that "service" at an arbitrary location within the computer network, use it for a defined time period, and then continues with its own program sequence. Client programs run on a connected network "resource".
- Service: an instantiation of a general application function, embodied in a computer program.
- Service Copy: one instance of the "service". The service copy (SC) is hosted on a "resource" computer, which provides both storage space and bandwidth for the access of the service.
- Resource (R): a host computer, which provides a limited number of storage space and access bandwidth for service transmission. Resources are connected via network connections defined in a topology script.
- Network Connections: These connections are intended to be of equal length and thus of equal transmission time and costs.

The trace collection of the simulation execution is done via a database for processing at a later stage after the simulation.

3.2. Message Flows in the Baseline Model

In order to simulate different control mechanisms we first consider the baseline system as a special case of the generic catallactic control mechanism. Through configuration in input scripts of the simulator, different behavior of the simulator can be set up. As a consequence, the comparison of simulation results should become easier to control and the development efforts focus on a single, generic system.

The centralized baseline mechanism employs a dedicated service coordinator (the master service copy, MSC), which is known to the individual service copies.

The client broadcasts a "request_service" message on its network connections. Either the receiving resource (R) provides a service copy (SC) of the requested type or not.

If a SC is available, the resource routes the request to that service copy, adding its costs for storage and bandwidth consumption. The SC directs the request to the Master Service Copy (MSC), provided with information about costs and the amount of the message's hop counter, i.e. the number of passed resources, indicating the distance to the requesting client.

Resource hosts (R) forward the received request – independent of the successful detection of the service – to their neighboring resource hosts, increasing the message's hop counter. Using this procedure, all adjacent resources will be inquired. If the hop counter exceeds a given number, the message is discarded.

The MSC receives all the information from the R/SC pairs, is able to compute the costs of providing a service and sends back an accept/propose message revealing the "cheapest" SC to the client. In addition, it informs the selected R/SC pair. The resource allocates a timeslot and the SC provides the service. After that, the client sends the formerly agreed reward to the SC, which redirects the payment share for bandwidth and storage to its R host.

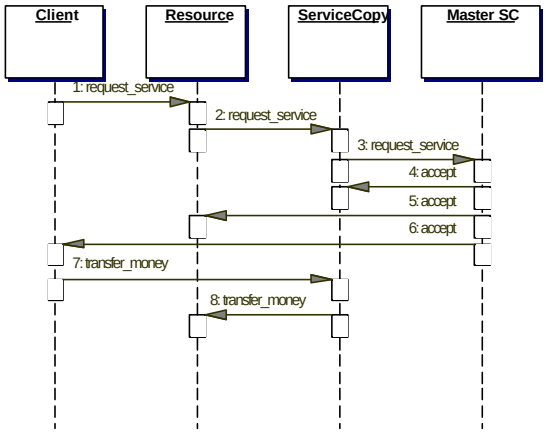

Figure 1. Money and Message Flows: Baseline Approach

3.3. Message Flows in the Catallactic Model

The Catallactic control mechanism has the characteristic that its resource allocation decisions are carried out by decentralized SCs with only local information about the environment.

Again, the clients send out a "service_request" message on its network connections in a Gnutella-like fashion [1; 14]. The receiving resource forwards the

message to the neighboring resource hosts. If the resource holds a SC of the requested type, the resource routes the request to it. In order to return a valid quote to the client, the SC has to inquire the resource about the provisioning costs by initiating a negotiation for bandwidth costs. A successful negotiation on this behalf allows the SC then to negotiate for the price for the provision of the service with the client.

The client orders all incoming proposals in its inbox and subsequently negotiates for service access. It is guided in its strategy by the subjective market price, which is computed from all price quotes the agent receives from the SCs, regardless of the particular sender. If the initial offer price does not match within an interval around the market price, the negotiation will be discontinued. Otherwise, the agents will engage in a bilateral alternating offers protocol [15] until acceptance or final rejection of the offer.

An accept message from the client lets the SC confirm both negotiations (with the resource for bandwidth and with the client for service provision). The resource reserves bandwidth and the contracts are sealed. The service provision is mirrored by the according money flow. On the other hand, a reject message from the client immediately stops further negotiation and initiates a reject message from the SC to the resource.

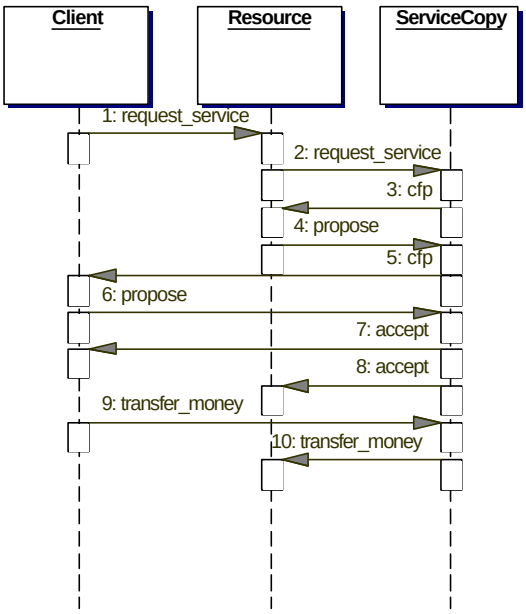

Figure 2. Money and Message Flows: Catallactic Approach

To maximize utility, the agents will change their initial offer prices, starting with demand and supply prices given in an input data script, according to the following scheme: Rs and SCs as sellers will lower their offer price by one money unit if the negotiation was not successfully finished. They will raise their initial price by one money unit after an offer has been accepted. The clients and SCs as buyers will change their initial prices vice versa.

If a SC has been turned down several times (having sent propose messages but never received an "accept"), it will try to relocate to another resource. According to the major share of received request messages, measured by incoming connections, the SC will ask the neighboring resource host for a free storage slot. If that target resource is fully occupied, the SC will ask the second-often relay of request messages and so on. If successful, the SC initializes a new instance at the target resource host and deletes the old instance. The overall effect is that SCs move themselves around the network in the physical direction of the demand. In the baseline approach, the SC wanting to relocate sends a query message to the MSC, who will inform the SC about where to relocate to.

4. Conducting Experiments

The application layer network is build on top of a physical network topology. The physical network topology is specified in the input of the simulator. The topology could be random or having a determined structure specified by the user. In Figure 3 we show one of the physical topologies, which we used in the experiments. This topology uses a central ring of nodes. On each central node, another ring of nodes is attached. Each of the attached nodes has a certain number of leaves.

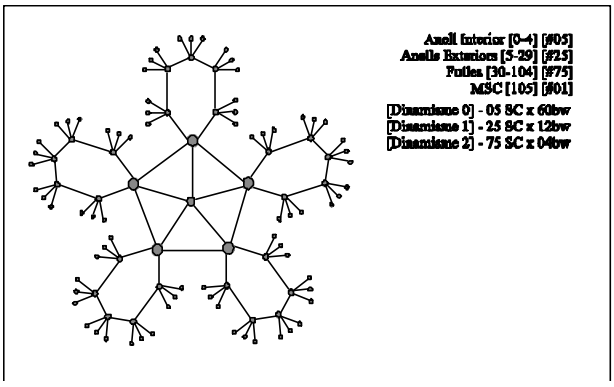

Figure 3. Example of network topologies, approx. 100 nodes

On top of the physical nodes, a number of different software agents are created, which form the application layer network. The software agents are Clients, Service Copies, and Resources. Each node can host Clients, Resources, and/or Service Copies. A node can host several agents or none at all. In the latter case, the node just acts as a router.

The application layer network formed by these agents is varied in the experiments. In order to simulate the node density of the application layer network, we vary the

number of Resource agents and Service Copies available to Clients. In order to simulate the dynamics of the application layer network, we connect and disconnect during the simulation the available Service Copies.

For each agent, particular data such as the capacity of the Resources can be specified in the initialization of the simulator. Recall that the capacity of the resources is high in the low node density scenario, and low in the high node density scenario, due to the correspondence with content distribution networks and peer-to-peer networks, respectively. The initial prices of Clients, Service Copies, and Resource agents are specified in the initialization of the simulator, also the initial budget of Clients.

The type of control mechanism is another parameter specified in the setup of the simulator. The main control mechanisms implemented in the simulator are the Catallactic and the baseline approach. The modular design of the simulator, however, also allows testing variations of them to investigate the effect of different parameters in each control approach.

Real world distributed applications like multimedia content distribution networks (for instance Akamai), Grid implementations, and Peer-to-Peer systems (for instance Gnutella) can be characterized in a simplified form by a number of a few common features, which inspired the design of the application layer network implemented in the simulator. Though different in many particular mechanisms, these real world applications can be mapped to the two-dimensional design space given by 1) the node dynamics; and 2) the node density of the application layer network.

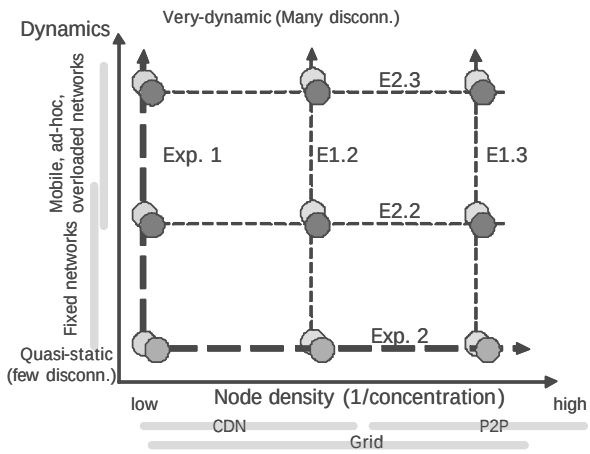

Figure 4: Illustration of the main experiments in a two-dimensional design space.

Node dynamics measures the degree of availability of service-providing nodes in the network. Low dynamics mean an unchanging and constant availability; high dynamics are attributed to a network where nodes start up and shut down with great frequency.

Node density measures the relation of resource nodes to the total number of network nodes. The highest density occurs when every network node provides the described service to others; the lowest density is reached if only one resource node in the whole network exists.

By varying node dynamics from null to medium to high, and node density from low to medium to high, each of the two control mechanism is simulated with 9 scenarios, as illustrated in Figure 4, which leads to 18 basic experiments.

The obtained traces are used to compare the performance of the Catallactic and the baseline system in the different scenarios. As an example, the following preliminary output measuring SWF could be found in an experiment varying the node density under a high dynamics regime (see Figure 5) (however, the results have not been statistically validated yet). This experiment was conducted with a demand trace of 2000 requests over 500s coming from 75 clients. In the low node density configuration (_20), the ALN consisted of 5 Service Copies and 5 Resources, in the medium node density configuration (_21) of 25 Resources and 25 Service Copies, and in the high node density configuration (_22) the application layer network was formed by 75 Resources and 75 Service Copies.

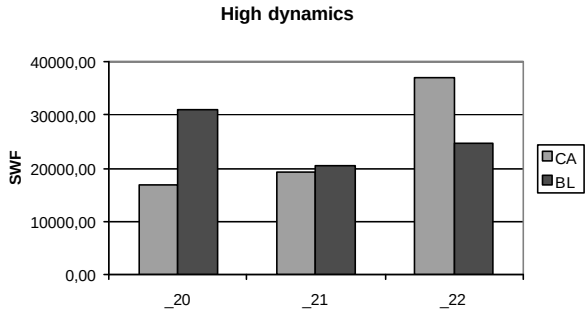

Figure 5: SWF comparison in a high node dynamics experiment with varying node density

One can see that with low node density (_20), the SWF in the Catallactic (CA) experiment is much lower than in the baseline (BL) case. With medium (_21) and especially high density (_22), the relations turn until the self-organizing Catallaxy outperforms the centralized baseline mechanism.

5. Conclusion and Outlook

CATNET is a network simulator for ALN which can simulate different resource allocation models. This article shows how, with a relatively simple variation of the negotiation protocol, both centralized and decentralized coordination can be supported, so that comparable results are produced. In our view, this feature distinguishes CATNET from most other Grid projects.

Although the main simulation parameters which we vary are the node dynamics and node density, we observed that actually the design space which could be considered for both systems is much larger. Other parameters we have considered to evaluate are, for instance, the effect of scale on the coordination mechanisms, the influence of particular characteristics of the demand trace, design parameters of the baseline system to handle highly dynamic environments, and parameters of the strategy used in the Catallactic coordination to determine prices.

References

[1] Adar, E. and B.A. Huberman, "Free Riding on Gnutella". *First Monday*, 5, 10 (2000), http://www.firstmonday.dk/issues/issue5_10/.

[2] Anderson, D.P. and J. Kubiatowicz, "The Worldwide Computer". *Scientific American*, 286, 3 (2002), pp. 28-35. http://www.cs.berkeley.edu/~kubitron/papers/.

[3] Breslau, L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation". *IEEE Computer*, 33, 5 (2002), pp. 59-67. http://ceng.usc.edu/~helmy/vint-computer-mag-article.pdf.

[4] Buyya, R., *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. Ph.D. Thesis. Monash University, Melbourne, Australia, 2002. http://www.buyya.com/thesis/thesis.pdf.

[5] Clearwater, S.H. *Market-based control. A paradigm for distributed resource allocation*. World Scientific, Singapore, 1996.

[6] Eymann, T., "Co-Evolution of Bargaining Strategies in a Decentralized Multi-Agent System". *AAAI Fall 2001 Symposium on Negotiation Methods for Autonomous Cooperative Systems*. Technical Report FS-01-03, AAAI, Falmouth, MA, 2001. pp. 126-134.

[7] Eymann, T. and B. Padovan, "The Catallaxy as a new Paradigm for the Design of Information Systems". *Proceedings of The World Computer Congress 2000 of the International Federation for Information Processing*. Beijing, China, 2000. http://citeseer.nj.nec.com/eymann00catallaxy.html

[8] Foster, I. and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit". *International Journal of Supercomputing Applications*, 11, 2 (1997), pp. 115-129.

[9] Frey, J., T. Tannenbaum, M. Livny, I.T. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids". *Cluster Computing*, 5, 3 (2002), pp. 237-246. http://www.globus.org/research/papers/condorg-hpdc10.pdf.

[10] Hayek, F.A., W.W. Bartley, P.G. Klein, and B. Caldwell. *The collected works of F.A. Hayek*. University of Chicago Press, Chicago, 1989.

[11] Huberman, B.A. *The Ecology of Computation*. North-Holland, Amsterdam, 1988.

[12] JavaSim Project. "JavaSim". *Ohio State University EEng Dept.* http://www.javasim.org/, last access 2003-02-28.

[13] Pruitt, D.G. *Negotiation behavior*. Academic Press, New York, 1981.

[14] Ripeanu, M., *Peer-to-Peer Architecture Case Study: Gnutella Network*. University of Chicago, Chicago 2001. http://www.cs.uchicago.edu/~matei/PAPERS/gnutella-rc.pdf

[15] Rosenschein, J.S. and G. Zlotkin. *Rules of encounter - designing conventions for automated negotiation among computers*. MIT Press, Cambridge, 1994.

[16] Sandholm, T.W. *Negotiation Among Self-Interested Computationally Limited Agents*. Ph.D. Thesis. University of Massachusetts, Amherst, 1996.

[17] Smith, R.E. and N. Taylor, "A Framework for Evolutionary Computation in Agent-Based Systems". *Proceedings of the 1998 International Conference on Intelligent Systems*. ISCA Press, 1998. http://www.ics.uwe.ac.uk/~rsmith/fecabs.pdf.

[18] Tesfatsion, L., "How economists can get alife". Arthur, W.B., Durlauf, S., and Lane, D.A. (eds.). *The Economy as a Evolving Complex System II*, pp. 533-564 Addison Wesley, Redwood City, CA 1997.

[19] Waldo, J., "The Jini Architecture for Network-centric Computing.". *Communications of the ACM*, (1999), pp. 76-82.