Decentralized vs. Centralized Economic Coordination of Resource Allocation in Grids

T. Eymann[1], M. Reinicke[1], O. Ardaiz[2], P. Artigas[2], L. Díaz de Cerio[2], F. Freitag[2], R. Messeguer[2], L. Navarro[2], and D. Royo[2]

[1] Institute for Computer Science and Social Studies, Albert-Ludwigs-University, Freiburg, Germany
{eymann, reinicke}@iig.uni-freiburg.de
[2] Computer Architecture Department, Polytechnic University of Catalonia, Barcelona, Spain
{oardaiz, partigas, ldiaz, felix, messeguer, leandro, royo}@ac.upc.es

Abstract. Application layer networks are software architectures that allow the provisioning of services requiring a huge amount of resources by connecting large numbers of individual computers, like in Grid or Peer-to-Peer computing. Controlling the resource allocation in those networks is nearly impossible using a centralized arbitrator. The network simulation project CATNET will evaluate a decentralized mechanism for resource allocation, 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 setup of money and message flows both for centralized and decentralized coordination in comparison.

1 Decentralized Resource Allocation Mechanisms and the Grid

Private computer centers, shielded from public networks, provide computation and data storage as a closed, private resource, and are mostly controlled by central arbitrator objects. In contrast, the openly accessible Internet resource pool offers more than 150 million connected hosts, and the number is growing exponentially, without any visible control instance. Even if only a fraction of this processing and storage capacity could be allocated properly, the resulting computation power would exceed private networks by far. There are additional advantages as well [2]:

- The system would be self-maintaining: if a computer is damaged the owner is responsible for repairing it; if the resource stays damaged another computer can take over its duties.
- Distributed data would be available from any location in the world and can probably survive disasters more securely than data stored on a single resource or network. Local catastrophes cause only local effects.

– The costs of using the network would be only a fraction of the costs compared to maintaining own hardware with frequent idle times. Enterprises always have vast capabilities on their disposal, but only pay for the time they actually need it.

Currently there exist some Internet -wide public resource infrastructures, which are called Grids and Peer-to-Peer systems. Grids are Internet accessible computational resources provided to grid users for execution of computational intensive parallel applications. Peer-to-Peer systems are end-users computer connected to the Internet which provide their computational and/or storage resources for other end-users usage [20]. Applications which can take advantage of provisioning of such huge amounts of resources are: multicast services for global audiences, storage repositories of peta-scale data sets, or parallel computational application. Such applications are executed in multiple resource locations distributed throughout the Internet. So that all those application instances work coordinately, they need to be organized forming a network on top of the Grid, therefore the name Application Network.

A Grid Application Network scenario would be the distributed provisioning of web services for Adobe's Acrobat (for creating PDF files) in an Akamai-like application layer network; word-processor client programs would transparently address the nearest/cheapest Acrobat service instance. The overall objective in the network would be (a) to always provide access to Acrobat service, such that a minimum number of service demands have to be rejected, and (b) to optimize network parameters such as provisioning costs and network communication.

In order to keep such a network operational, service control and resource allocation mechanisms are required. However, these mechanisms are realized in existing operational large-scale distributed systems by employing a centralized coordinator instance (like an auctioneer or an arbitrator). This centralized approach has several drawbacks.

A first prerequisite for a central coordination instance to work properly is that the environment does not change its state between the beginning and the end of the computation process, e.g. by "sliced" computing in discrete timeslots. Grid application networks, however, are very dynamic and fast changing systems: service demands and nodes connectivity changes are very frequent, and new different services are created and composed continuously. Dynamic grid application networks need a continuously updating 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 grid application networks 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 and centralized 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 optimised behavior with regard to low overhead communication, short computation times, pareto-optimal resource allocation. In addition to that, the coordination concept should avoid the so-called over-usage of shared resources – known as the "tragedy of commons" [11] – or "free- riding behavior" [1], [13], [28], which can lead to the network's collapse.

Recent research in Grid computing has also recognized the value of price generation and negotiation, and in general 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. Examples are the Nimrod/G Resource Broker and the GridBus project [5], [10].

As a free-market economy is able to adjudicate and satisfy the conflicting needs of millions of human agents [16], it would be interesting to evaluate if this decentralized organizational principle could also be used for coordination of grid application networks. 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 describes money and message flows in the grid application network economic model, both with a centralized (baseline) and a decentralized implementation. The article closes with some preliminary results and an outlook on the applicability of the concept to various domains.

2 Decentralized Economic Coordination: the Catallaxy Paradigm and the CATNET Project

In grid application networks, different types of resources can be scarce such as storage, bandwidth, and CPU cycles. Optimization criterions for allocating these resources can be based on cost-efficiency, performance or a combination of parameters. In this work, our goal is to develop a simulator, which allows to experimentally compare two main resource allocation strategies: A centralized approach in which decisions are taken centrally and a decentralized approach, where local agents negotiate resources using economic models.

The Catallaxy coordination approach [7], [12] is a coordination mechanism for systems consisting of autonomous decentralized hard- or software devices, which is based on constant negotiation and price signaling between the devices. The mechanism is based on efforts from both agent technology and economics, namely agent-based computational economics [27], to develop new technical possibilities

of coordinating decentralized information systems consisting of autonomous software agents. The software agents are able to adapt their strategies using machine learning mechanisms [26], 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 [7]. The resulting patterns are comparable to those witnessed in human market negotiation experiments [19].

Earlier work in computer science has used economic principles for resource allocation in operating systems, packet routing in computer networks, and load balancing in distributed computer systems [6], [14]. 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.

The goal of the CATNET project is thus to evaluate the Catallaxy paradigm for decentralized operation of grid application networks in comparison to a baseline centralized system. For the evaluation of the overall success of the control mechanism, we will use the "maximum social welfare-criterion", which is the sum of all utilities of the participating nodes [23]. This criterion balances both costs and revenue incurred by the nodes and allows comparing different variants of the Catallaxy and baseline implementations.

Social welfare maximizing solutions are a subset of "Pareto-efficient" ones; once the sum of the payoffs is maximized, an agent's payoff can increase only if another agent's payoff decreases [29]. The resource allocation efficiency of an agent adds to the revenue, while communication cost, measured as the ratio of data to control bandwidth consumption, adds to the costs. Increasing performance and decreasing communication in the whole network thus directly computes to relatively maximize social welfare. As this property also holds for local optima of the solution space, "social welfare" is considered to be the main, but not the only evaluation parameter. Other evaluation parameters will be the network traffic and service access latency.

3 Money and Message Flows in the Grid and Application Network

The lifecycle of a grid application network can be divided in two phases, the deployment and the allocation phase.

The goal of the deployment phase is the initial positioning of new resources, services, and service copies [3]. We assume that the deployment phase has already been carried out and services are initially located in the network. Deployment can also be economically modeled, as self-interested service deployers compete for existing resources where services are to be placed, and utility-maximizing resource providers compete for the provisioning of promising new services.

The allocation phase, which is in the main focus here, changes resource allocations during the runtime of the network, meaning a re-allocation of the initial

positions found in the deployment phase. 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 CATNET Network Simulator

CATNET is a simulator for a generic grid application network (GAN). This GAN simulator is implemented on top of the JavaSim network simulator. It can be configured to simulate a specific GAN, 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.

JavaSim is a component-based, compositional simulation environment [4], [15]. It is a discrete event simulator targeted at networking research that provides support for simulation of real network topologies and grid application services, i.e. data and control messages among application network instances.

JavaSim has been built upon the notion of the autonomous component programming model. Similar to COM/COM+, JavaBeans, or CORBA, the basic entity in JavaSim are components, but unlike the other component- based software packages/standards, components in JavaSim are autonomous. Having been developed entirely in Java, reusing the code has been easy.

For the purpose of network modeling and simulation, the model defines on top of the autonomous component architecture a generalized packet switched network model. 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.

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: a host computer, which provides a limited number of storage space and access bandwidth for service transmission. Resources (R) are connected to each other via dedicated network connections.
– 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, 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.

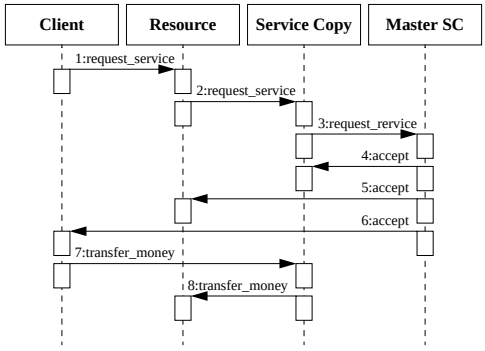

Fig. 1. Money and Message Flows: Baseline Approach

As Fig. **??** shows, 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.

Contracts have to be fulfilled; a re-negotiation of allocations is out of the project's scope. Right after the service has been provided to the client, the client sends the formerly agreed reward to the SC, which redirects the payment share for bandwidth and storage to its R host.

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; see Fig. **??**.

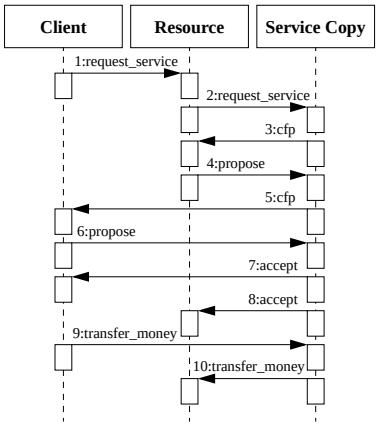

Fig. 2. Money and Message Flows: Catallactic Approach

Again, the clients send out a "service_request" message on its network connections in a Gnutella-like fashion [1], [20]. 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 allows the SC then to negotiate for the price for the provision of the service with the client, like in a very shallow iterated contract net protocol [9], [25].

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 gets "from the market", 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 [21] 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.

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 Conclusion and Outlook

One of the goals of the CATNET project is the setup for a network simulator which can simulate different coordination models. This article shows how centralized and decentralized coordination can be supported with a relatively simple addition to the negotiation protocol, so that comparable results are produced. The findings can be visualized using NAM [17];

The final evaluation whether the baseline or catallactic mechanism receives better results has not been made yet. This result will be achievable in the last project phase in spring 2003. For the time being, the CATNET simulator in itself already allows investigation into allocation and messaging behavior in grid application networks.

If CATNET is successful with regard to the Catallactic control mechanism, allocation and scheduling questions in other decentralized network domains like

hospital logistics [22], factory logistics [18] or adaptive supply chain management [8], [24] could also be targeted.

In our view, CATNET stands at the very beginning of research into Catallactic Information Systems. In Fig. ??, we have indicated how future research work can be divided into the agent technology layer and an application-specific layer. Both are linked in a feedback loop. On one hand, the technology has to constantly (and imperfectly) model an ever-changing state of the application world. On the other hand, technology's results and the behavior of its single elements directly influence the application state by means of self-organization.

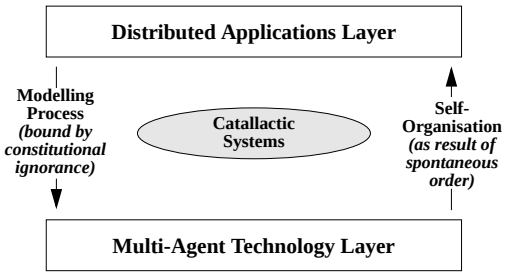

Fig. 3. Catallactic Information Systems

Future research will address the design of control institutions for large, open, and heterogeneous agent societies. These institutions should influence the multi-agent systems to enable them to emergently develop towards a state of desirable global behavior where security, trust and welfare are provided to all participants. Our research and software is still in its early infancy, but we hope to be able to provide a first "proof of concept" for Catallactic Information Systems in the domain of decentralized grid application networks.

References

1. Adar, E., Huberman B.A.: Free Riding on Gnutella. First Monday, 5, 10, 2000. http://www.firstmonday.dk/issues/issue5_10.
2. Anderson, D.P., Kubiatowicz, J.: The Worldwide Computer. Scientific American, 286, 3, 28-35, 2002. http://www.cs.berkeley.edu/ kubitron/papers.
3. Ardaiz, O., Freitag, F., Navarro, L.: Multicast Injection for Application Network Deployment. 26nd IEEE Conference on Local Computer Networks.Tampa, Fla., 2001.
4. Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., Yu H.: Advances in Network Simulation. IEEE Computer, 33, 5, 59-67, 2002. http://ceng.usc.edu/ helmy/vint-computer-mag-article.pdf.
5. Buyya, R., Abramson D., Giddy J.: A Case for Economy Grid Architecture for Service-Oriented Grid Computing. 10th IEEE International Heterogeneous Computing Workshop (HCW), 2001.

6. Clearwater, S.H.: Market-based control a paradigm for distributed resource allocation. World Scientific, Singapore, 1996.
7. Eymann, T.: Co-Evolution of Bargaining Strategies in a Decentralized Multi-Agent System. AAAI Fall 2001 Symposium on Negotiation Methods for Autonomous Cooperative Systems, 2001.
8. Eymann, T., Padovan, B.: The Catallaxy as a new Paradigm for the Design of Information Systems. Proceedings of The World Computer Congress of the International Federation for Information Processing, 2000.
9. FIPA: FIPA Specification. FIPA Website. http://www.fipa.org.
10. Grid Computing and Distributed Systems (GRIDS) Laboratory: GRIDBUS Project. The University of Melbourne, Australia. 2002. http://www.gridbus.org.
11. Hardin, G.: The Tragedy of the Commons. Science, 162, 1243-1248, 1968.
12. Hayek, F.A., Bartley, W.W., Klein, P.G., Caldwell, B.: The collected works of F.A. Hayek. University of Chicago Press, Chicago, 1989.
13. Hogg, T., Huberman, B.A.: Controlling Chaos in Distributed Systems". IEEE Transactions on Systems, Man and Cybernetics, 21, 6, 1325-1332, 1991.
14. Huberman, B.A.: The Ecology of computation. North-Holland, Amsterdam, 1988.
15. JavaSim Project: JavaSim. Ohio State University, EEng Dept. http://www.javasim.org.
16. Kephart, J.O., Hanson J.E., Grosof B.N., Sairamesh J., White S.R.: Dynamics of an information filtering economy. Klusch, M. and Weiss, G. (eds.). Lecture Notes in Artificial Intelligence, 160-171 Springer, Heidelberg, 1998.
17. NS/NAM Project: NAM Network Animator. USC Information Sciences Institute, 2002. http://www.isi.edu/nsnam/nam/index.html
18. Parunak, H.V.D.: Industrial and Practical Applications of Distributed Artificial Intelligence. Wei, G. (ed.). Multi-Agent Systems, 377-457 The MIT Press, Cambridge, MA, 1999.
19. Pruitt, D.G.: Negotiation behavior. Academic Press, New York, 1981.
20. Ripeanu, M.: Peer-to-Peer Architecture Case Study: Gnutella Network. University of Chicago, Chicago 2001.
21. Rosenschein, J.S., Zlotkin, G.: Rules of encounter - designing conventions for automated negotiation among computers. MIT Press, Cambridge, 1994.
22. Sackmann, S., Eymann, T., Mller, G.: EMIKA - Real-Time Controlled Mobile Information Systems in Health Care Applications. Workshop "Mobiles Computing in der Medizin" im Rahmen der 7. Fachtagung "Praxis der Informationsverarbeitung in Krankenhaus und Versorgungsnetzen".Heidelberg, 2002.
23. Sandholm, T.W.: Negotiation Among Self-Interested Computationally Limited Agents. University of Massachusetts, Amherst, 1996.
24. Shepherdson, J.W., Thompson, S.G., Odgers, B.R.: Decentralised Workflows and Software Agents. BT Technology Journal, 17, 4, 65-71, 1999.
25. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers, 29, 1104-1113, 1980.
26. Smith, R.E., Taylor N.: A Framework for Evolutionary Computation in Agent-Based Systems. International Conference on Intelligent Systems, ISCA Press, 1998.
27. 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, 533-564. Addison Wesley, Redwood City, CA, 1997.
28. Thomas, J. and Sycara, K.: Heterogeneity, Stability and Efficiency in Distributed Systems. Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98), Paris, France, 1998.
29. Varian, H.R.: Intermediate Microeconomics. W.W. Norton, New York, 1999.