

EXPLORING DECENTRALIZED RESOURCE ALLOCATION IN APPLICATION LAYER NETWORKS

Torsten Eymann, Michael Reinicke
Institute for Computer Science and Social Studies
Albert-Ludwigs-University Freiburg, Germany
E-mail: {eymann,reinicke}@iig.uni-freiburg.de

Oscar Ardaiz, Pau Artigas, Luis Díaz de Cerio, Felix Freitag,
Roc Messeguer, Leandro Navarro, Dolors Royo
Computer Architecture Department
Polytechnic University of Catalonia, Spain
E-mail: {oardaiz,partigas,ldiaz,felix,
messeguer,leandro,royo}@ac.upc.es

KEYWORDS

Decentralized economic coordination, Application networks

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. Controlling the resource allocation in those networks is nearly impossible using a centralized arbitrator. Decentralized concepts have been proposed for peer-to-peer networks. Achieving the network functionality, however, is often more important than reducing its overall cost. We propose a decentralized mechanism for resource allocation in application layer networks based on the economic paradigm of the Catallaxy, against a centralized mechanism using an arbitrator object. In both approaches, software agents buy and sell network services and resources to and from each other. We describe the main ideas of our approach and the implementation of the CATNET simulator for application layer networks to evaluate the both centralized and decentralized coordination.

1. INTRODUCTION

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. Depending on the mechanism used for allocation of processing and storage capacity, a number of interesting advantages could be achieved (Anderson and Kubiatawicz 2002):

- Computation power would exceed private networks by far, even if only a fraction of the network's capacity could be allocated properly.
- 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.

Application layer networks are software architectures that aims at allowing the provision of services requiring a huge amount of resources by connecting large numbers of individual computers for information search, content download, parallel processing or data storage. These networks are built on the application layer on top of the existing TCP/IP Internet protocol.

Application layer networks are needed, for instance, to set up multicast services for large-scale global audiences, to provide services for storing ultra large-scale data sets, and to allow the execution of parallel applications requiring teraflops of processing power. Common concepts are GRID computing (mostly for distributed processing) and Peer-to-Peer-Computing (mostly for distributed data storage and access).

In this paper we explore the application of the economic paradigm of the Catallaxis to the resource allocation in application layer networks. Our approach applies a fully decentralized concept, in which decisions are made by independent agents targeting their local optimization.

The following section 2 outlines the problems found with centralized coordination in dynamic environments. In section 3 we propose the decentralized economic coordination based on the Catallaxis. Section 4 shows application scenarios and describes the implementation of the CATNET simulator for application network evaluation. We show the money and message flows used in the economic models, both in the centralized and the decentralized case. And outline design of the experiments. Section 5 concludes with an outlook on the extension of the concept to various domains.

2. PROBLEMS IN CENTRALIZED RESOURCE ALLOCATION

In order to keep an application network operational, service control and resource allocation mechanisms are required.

Often, these mechanisms are operated by employing a centralized coordinator instance, like an auctioneer or an arbitrator. Such a 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. Application layer 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 application layer 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 application layer 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.

3. DECENTRALIZED ECONOMIC COORDINATION: THE CATALAXY PARADIGM

The drawbacks found in the centralized approach 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 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” (Hardin 1968) – or “free-riding behavior” (Adar and Huberman 2000, Ripeanu 2001), which can lead to the network’s collapse.

Application layer networks build by loosely connected nodes are dynamic systems, where changes in the service demand and in the node connectivity are usual. New services are

created and composed continuously. Nodes forming the network connect and disconnect frequently. Another characteristic is that objects exchanged like music clips are mainly “small” objects. In future systems, however, the content can be of any form, including audio, video, and large data sets. To achieve an efficient performance of such systems, more intelligent decisions than we have in today’s systems are required, concerning how service provision in such networks can be coordinated, from where the content should be retrieved and on which path it should travel.

As a free-market economy is able to adjudicate and satisfy the conflicting needs of millions of human agents (Kephart et al. 1998), it is interesting to evaluate if this decentralized organizational principle could also be used for coordination of application layer networks.

The Catalaxy coordination approach (Eymann et al. 2000, Eymann 2001, Hayek 1989) 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, leading to agent-based computational economics (Tsfatsion 1997), 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 (Smith and Taylor 1998), and this constant revision of strategies leads to a evolution of software agent strategies, a stabilization of prices throughout the system and self-regulating coordination patterns (Eymann et al. 2000). The resulting patterns are comparable to those witnessed in human market negotiation experiments (Pruitt 1981).

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 (Clearwater 1996). 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 Catalaxy 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.

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 (Buyya et al. 2001, Gridbus Project 2002).

In application layer networks, different types of resources can be scarce such as storage, bandwidth, and CPU cycles. Optimization criteria 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 goal of the CATNET simulator is to evaluate the Catallaxy paradigm for decentralized operation of application layer 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 (Sandholm 1996). 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 (Varian 1999). 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.

4. SIMULATION OF CATALLECTIC NETWORK COORDINATION

4.1 Application scenarios

The lifecycle of an application layer 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. This paper assumes that the deployment phase has already been carried out and services are initially located in the network. In principle, 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.

Real world distributed applications like multimedia content distribution networks (for instance Akamai [1]), Grid implementations, and Peer-to-Peer systems (for instance Gnutella) are target applications for the CATNET simulator. In the design of the application layer network implemented in the simulator, we considered that although different in many particular mechanisms, these real world applications can be characterized in a simplified form by a number of a few common features, which we identify by 1) the node dynamics; and 2) the node density of the application layer network.

4.2 The CATNET simulator

We have implemented the CATNET simulator for a generic application layer network. This simulator is implemented on top of the JavaSim network simulator. It can be configured to simulate a specific application network, 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 (JavaSim Project 2002). It is a discrete event simulator targeted at networking research that provides substantial support for simulation of real network topologies and application layer 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 is straightforward.

For the purpose of network modeling and simulation, JavaSim 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 simulator implements two main control mechanisms for network coordination: the baseline control mechanism and the 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.

4.2 Network topologies

Application layer networks are formed on top of a physical network. The physical network topology is specified in the input of the simulator. The topology can be random and generated by a network topology generator, or having a determined structure specified by the user.

Figure 1 illustrates a small physical topology. 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.

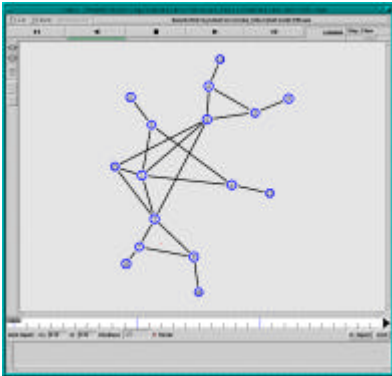


Figure 1. Examples of network topologies with 17 nodes.

4.3 Application layer network

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

- **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 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.

In the simulator, we consider the service as the functionality of the application layer network, which is required by clients and provided by service and resource agents. The concept of service can be described in the following way:

- **Service:** an instantiation of a general application function, embodied in a computer program.

A concrete case for an application is for instance, the distributed provisioning of web services for Adobe’s Acrobat (for creating PDF files) in an Akamai-like application layer network. Here, word-processor client programs would transparently address the nearest/cheapest Acrobat service instance in order to create PDF files. 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.

4.3 Money and message flows

In the simulator, the network activity is characterized by a continuous exchange of control messages and service provision. Different control messages in the two coordination mechanisms are used to accomplish the negotiation between the agents.

In Figure 2 we show the money and message flow used in the catalytic coordinated system. The requests from clients are broadcasted and forwarded to the appropriate ServiceCopies. Compared with the flooded requests model used in Gnutella (Ripeanu 2001), however, the numbers of hops a request can be forwarded in the simulator is limited. ServiceCopies initiate negotiations with the hosting Resource to provide the service. Upon successful negotiations the ServiceCopies offer the service to the Client. If the Client accepts, the ServiceCopy provides the service to the Client by means of a Resource.

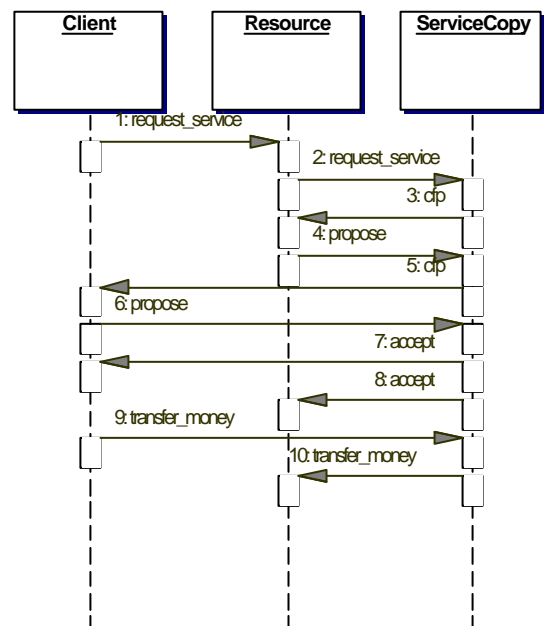


Figure 2. Money and message flow in the Catalytic approach.

In the centralized baseline system (Figure 3), the MasterServiceCopy (MSC) receives the Client requests with additional information through the Resource/ServiceCopy pairs. Taking into account the distance and availability, it selects a Resource/ServiceCopy pair and sends back an Accept or Reject message to the Client.

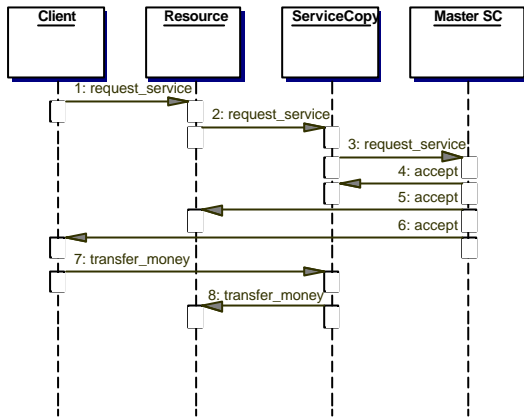


Figure 3. Money and message flow in the Baseline approach.

4.4 Input

The input of the simulator is on one hand the service demand trace, on the other hand all the data needed to set-up the network including the configuration of the physical network in terms of node topology, the application layer network concerning the agent layout, the parameters to configure the network dynamics, and the initial prices.

The service demand trace of the simulator contains the service requests from the clients. The clients specify the required service. The simulator allows the clients to detail the requested service through some additional parameters. The service demand trace can be generated by the simulator, which includes a tool to generate random demand traces. For controlled experiments, the demand trace can be generated manually.

4.5 Experiments

Our approach is to map different application layer networks from real world into a two-dimensional design space consisting of the node dynamics and node density. The range of the node density goes from low node density (content delivery networks) to high node density (peer-to-peer networks). The node density is set in the configuration of the simulator. The parameter, which governs the dynamics of the network, affects the connection and disconnection of ServiceCopies. The node density is given by the number of resource agents available in the network.

In Figure 4, the correspondence of node density and node dynamics to real world systems is depicted. As approximation, low density (which also means high resource concentration) may correspond to web service scenarios with

one or a few web servers. Medium density may correspond to a CDN or a Grid. High density (which also means spread resources) may correspond to a P2P network, or an extreme case of Grid.

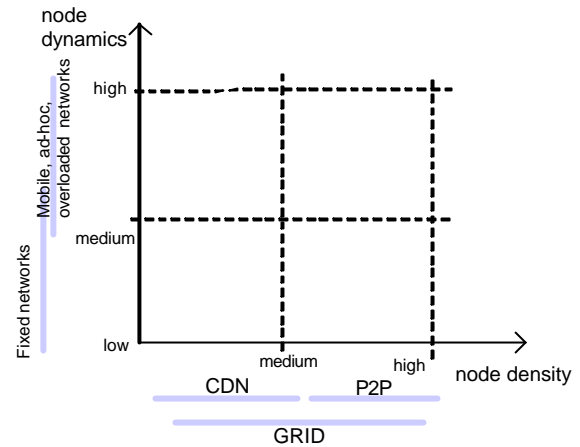


Figure 4. Correspondence of real application layer networks to the two-dimensional design space of the simulator.

5. CONCLUSIONS AND OUTLOOK

One of the goals of the CATNET project is the setup for a network simulator, which can simulate different coordination models. This paper shows how centralized and decentralized coordination can be supported with a relatively simple addition to the negotiation protocol.

Currently, the CATNET simulator allows investigation into allocation and messaging behavior in application layer networks. If CATNET is successful with regard to the Catalytic control mechanism, allocation and scheduling questions in other decentralized network domains like hospital logistics (Sackmann et al 2002), factory logistics (Parunak 1999) or adaptive supply chain management (Eymann and Padovan 2000) could also be targeted.

In our view, CATNET stands at the very beginning of research into Catalytic Information Systems. In Figure 5 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.

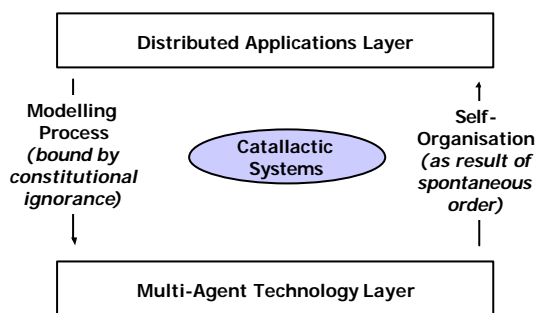


Figure. 5 Catalallactic Information Systems

Future research can 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 infancy, but we hope to be able to provide a first "proof of concept" for Catalallactic Information Systems in the domain of decentralized application layer networks.

ACKNOWLEDGEMENT

CATNET has been supported by European Commission Information Society Technologies Programme under contract no. IST-2001-34030.

REFERENCES

- Adar, E. and B.A. Huberman, "Free Riding on Gnutella". *First Monday*, 5, 10 (2000), http://www.firstmonday.dk/issues/issue5_10/
- Anderson, D.P. and J. Kubiatowicz, "The Worldwide Computer". *Scientific American*, 286, 3 (2002), 28-35. <http://www.cs.berkeley.edu/~kubitron/papers/>
- 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), 59-67. <http://ceng.usc.edu/~helmy/vint-computer-mag-article.pdf>
- Buyya, R., D. Abramson, and J. Giddy, "A Case for Economy Grid Architecture for Service-Oriented Grid Computing". *Proc. 10th IEEE International Heterogeneous Computing Workshop (HCW 2001)*. San Francisco, 2001. <http://www.csse.monash.edu.au/~davida/papers/ecogrid>
- Clearwater, S.H. *Market-based control paradigm for distributed resource allocation*. World Scientific, Singapore, 1996.
- 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.
- Eymann, T. and B. Padovan, "The Catalaxy as a new Paradigm for the Design of Information Systems".

Proceedings of The World Computer Congress 2000 of the International Federation for Information Processing. 2000.

- Gridbus: Grid Computing and Distributed Systems (GRIDS) Laboratory. "GRIDBUS Project". *The University of Melbourne, Australia*. <http://www.gridbus.org/>, 2002-11-28.
- Hardin, G., "The Tragedy of the Commons.". *Science*, 162, (1968), 1243-1248.
- 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.
- Hogg, T. and B.A. Huberman, "Controlling Chaos in Distributed Systems". *IEEE Transactions on Systems, Man and Cybernetics*, 21, 6 (1991), 1325-1332.
- Huberman, B.A. *The Ecology of computation*. North-Holland, Amsterdam, 1988.
- JavaSim Project. "JavaSim". *Ohio State University EEng Dept.* <http://www.javasim.org/>, 2 A.D.-11-29.
- Kephart, J.O., J.E. Hanson, B.N. Grosf, J. Sairamesh, and S.R. White, "Dynamics of an information filtering economy". Klusch, M. and Weiss, G. (eds.). *Lecture Notes in Artificial Intelligence*, 160-171 Springer, Heidelberg 1998.
- Parunak, H.V.D., "Industrial and Practical Applications of Distributed Artificial Intelligence". Weiß, G. (ed.). *Multi-Agent Systems*, 377-457 The MIT Press, Cambridge, MA 1999.
- Pruitt, D.G. *Negotiation behavior*. Academic Press, New York, 1981.
- 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>
- Sackmann, S., T. Eymann, and G. Müller, "EMIKA - Real-Time Controlled Mobile Information Systems in Health Care Applications". 2. *Workshop "Mobiles Computing in der Medizin" im Rahmen der 7. Fachtagung "Praxis der Informationsverarbeitung in Krankenhaus und Versorgungsnetzen"*. Heidelberg, 2002.
- Sandholm, T.W. *Negotiation Among Self-Interested Computationally Limited Agents*. University of Massachusetts, Amherst, 1996.
- Shepherdson, J.W., S.G. Thompson, and B.R. Odgers, "Decentralised Workflows and Software Agents". *BT Technology Journal*, 17, 4 (1999), 65-71
- Smith, R.G., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". *IEEE Transactions on Computers*, 29, (1980), 1104-1113.
- 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>
- Tesfatsion, L., "How economists can get alive". 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.