Information Society Technologies

IST-FP6-003769 CATNETS
Y2 Report
WP 1: Theoretical and Computational Basis

Contractual Date of Delivery to the CEC: 31. August 2006

Actual Date of Delivery to the CEC: 30. September 2007

Author(s): Daniel Veit, Georg Buss (University of Mannheim)
Björn Schnizler, Dirk Neumann (University of Karlsruhe)
Werner Streitberger, Torsten Eymann (University of Bayreuth)

Workpackage: WP 1 Theoretical and Computational Basis

Est. person months: 19

Security: public

Nature: submitted version

Version: 1.0

Total number of pages: 59

Abstract:

This report covers the results of WP1 in Y2 and hence comprises mainly the results from T1.4 and T1.5.

Keywords (optional):

Decentralized Market Mechanisms, Centralized Market Mechanisms, Catallaxy, Market Engineering, Simulator Integration, Prototype Integration

CATNETS Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-FP6-003769. The partners in this project are: LS Wirtschaftsinformatik (BWL VII) / University of Bayreuth (coordinator, Germany), Arquitectura de Computadors / Universitat Politecnica de Catalunya (Spain), Information Management and Systems / University of Karlsruhe (TH) (Germany), Dipartimento di Economia / Universitá delle marche Ancona (Italy), School of Computer Science and the Welsh eScience Centre / University of Cardiff (United Kingdom), Automated Reasoning Systems Division / ITC-irst Trento (Italy), Chair of Business Administration and Information Systems - E-Business and E-Government / University of Mannheim (Germany).

University of Bayreuth
LS Wirtschaftsinformatik (BWL VII)
95440 Bayreuth
Germany
Tel: +49 921 55-2807, Fax: +49 921 55-2816
Contactperson: Torsten Eymann
E-mail: catnets@uni-bayreuth.de

University of Karlsruhe
Institute for Information Management and Systems
Englerstr. 14
76131 Karlsruhe
Germany
Tel: +49 721 608 8370, Fax: +49 721 608 8399
Contactperson: Christof Weinhardt
E-mail: weinhardt@iism.uni-karlsruhe.de

University of Cardiff
School of Computer Science and the Welsh eScience Centre
University of Caradiff, Wales
Cardiff CF24 3AA, UK
United Kingdom
Tel: +44 (0)2920 875542, Fax: +44 (0)2920 874598
Contactperson: Omer F. Rana
E-mail: o.f.rana@cs.cardiff.ac.uk

University of Mannheim
Chair of Business Administration and Information Systems
- E-Business and E-Government -
L9, 1-2
68131 Mannheim
Germany
Tel: +49 621 181 3321, Fax: +49 621 181 3310
Contactperson: Daniel Veit
E-mail: veit@uni-mannheim.de

Universitat Politecnica de Catalunya
Arquitectura de Computadors
Jordi Girona, 1-3
08034 Barcelona
Spain
Tel: +34 93 4016882, Fax: +34 93 4017055
Contactperson: Felix Freitag
E-mail: felix@ac.upc.es

Universitá delle marche Ancona
Dipartimento di Economia
Piazzale Martelli 8
60121 Ancona
Italy
Tel: 39-071- 220.7088 , Fax: +39-071- 220.7102
Contactperson: Mauro Gallegati
E-mail: gallegati@dea.unian.it

ITC-irst Trento
Automated Reasoning Systems Division
Via Sommarive, 18
38050 Povo - Trento
Italy
Tel: +39 0461 314 314, Fax: +39 0461 302 040
Contactperson: Floriano Zini
E-mail: zini@itc.it

Contents

Chapter 1

Introduction

The primary target of the CATNETS project is the quantitative comparison between the technical and economic efficiency of market-based resource allocation mechanisms in application layer networks such as Grids. Here, two fundamentally different approaches are compared. A centralized – auction-based – market mechanism and a decentralized – Catallaxy-based – market mechanism.

After the reorganization (following the Y1 review) this endeavor has been approached in the following way:

- Workpackage 1 (Theoretical and Computational Basis): The target of this workpackage is the definition of market mechanisms for the centralized and the decentralized case. Therefore, software components have to be identified (T1.1), market requirements have to be analyzed (T1.2) and an architecture for services and ALNs has to be designed (T1.3). Finally a specification for bidding and interaction issues has to be carried out (T1.4). A specification and analysis of the market mechanisms concludes WP1.

- Workpackage 2 (Simulation Framework): The core of this workpackage is the implementation of a simulator framework integrating both, the centralized and the decentralized market mechanism. The goal is to compare the outcome of the application of both mechanisms quantitatively.

- Workpackage 3 (Proof-of-Concept Applications): In parallel to WP2 this workpackage focuses on the implementation of the designed decentralized market mechanisms into a prototypical ALN-middleware software. Quantitative evaluations are carried out by running experiments with this platform.

- Workpackage 4 (Performance Evaluation): The aim of this WP is the identification and design of metrics in order to measure the outcome of WP2 and WP3. Here, a metrics framework is designed in order to enable the measurement of the quality of allocation results in using an economic scale.

- Workpackage 5 (Management): This workpackage is designed to carry out project management and dissemination.

This report covers the results of WP1 in Y2 and hence comprises mainly the results from T1.4 and T1.5.

The remainder of this report is structured as follows: Chapter 2 illustrates research questions of the CATNETS project in the context of self-organizing systems and draws a vision towards future research topics. In the following, Chapter 3 focuses on the description of the introduced market mechanisms. In Section 3.1 the properties of the centralized market mechanisms, which have been defined in Y1 report [SNV+05b] are briefly described. Section 3.2 provides a formal description of the decentralized allocation mechanisms. The bidding issues, presented in Chapter 4, elaborate different scenarios connecting the service and resource market. Advantages and disadvantages of these scenarios are compared to enable a comparison of the centralized and decentralized market mechanism. In Chapter 5 the integration of the mechanisms into the OptorSim simulator is outlined and links to WP2 are set. Therein, preliminary simulation results are provided. The next chapter - Chapter 6 - shows the integration of the catallactic market into the middleware environment. Implementation details and first results from a scenario are presented there. Chapter 7 relates Workpackage 1 to the other workpackages. Finally, Chapter 8 provides a summary and an outlook an the work that is content in Y3.

Chapter 2

Self-Organization in Computing Systems - Putting CATNETS into a Greater Perspective

2.1 Introduction to Self-Organization

The vision of Self-Organization in Computing Systems and Networks has gained significant interest in the last years, and even was labeled with a popular buzzword: Autonomic Computing [KC03] describes a concept of self-organizing information technology, where the functionality of the computing system is an emergent feature of the capabilities and actions of the components. Without any centralized controller, this system is capable of configuring, healing, organizing and protecting itself (the so-called CHOP properties). In contrast, classical IT control involves a centralized controller instance with global knowledge about the current status of the computing system, and a detailed regulation mechanism to 'heal' deviations from a defined 'normal' status. Centralized computation is said to be not that flexible in terms of scalability and adaptability. Autonomic Computing uses a biological paradigm as a design and control metaphor, the autonomic nervous system. The core CHOP properties of the Autonomic Computing concept are intended to be an electronic realization of the respective mechanisms of the human body. Self-organization can be found in other parts of our natural environment as well, e.g. biological evolution, social group behavior, market dynamics phenomena and other complex adaptive systems. Autonomics refers to our own human neural system, Catallaxy [ESMP03] to self-organizing markets in Economics, Stigmergy to coordination without communication in insect colonies. All these ideas have in common that the solution for growing complexity both in scale and semantics is decentralized control, that is based on local information and executed through local effects which build up to a system-wide emergent behavior. It is not surprising that projects labeled Autonomic Computing are thus manifold, coming from diverse backgrounds and academic habitats, and aiming at a variety

of technological and scientific knowledge increase. This is because it is not only hoping to solve many problems that industries are experiencing, with unreliable or increasingly more complex environments. In addition, it brings a unique challenge to computer scientists and engineers to add the concept of the system's 'self'. Furthermore, it attracts a diverse community from biological systems science to operating systems engineering, who are often able to present solutions evolved from earlier work. Historically, Self-Organizing Computing clearly builds on the distributed artificial intelligence paradigm. Multiagent systems [Wei99] have thus gained renewed attention and a rewarding application area. In striking similarity to the evolution of Distributed AI itself, the key motivation aspect again lies in the increasing size and complexity of the information systems to be controlled, and a non-negligible growth in their control costs. Again, the concept of self-organizing computing is not new. Earlier concepts, like cybernetic [Wie98] or autopoietic [Mat99] systems, failed because of technological immaturity, missing business models or a combination thereof. It seems that technological progress now allows another, this time more promising, look. It is yet unclear whether Autonomic Computing will survive as its own topic in the long run. However, by stimulating creativity of researchers, recombining existing knowledge from different fields, spurring workshops, conferences and research proposals, Autonomic Computing is already a positive phenomenon in computing science research.

2.2 Infrastructural Spheres of Self-Organizing Computing

To get a clearer look at the prospects and hurdles, chances and risks of Self-Organizing Computing (SOC) the complex concept of SOC will be divided into three spheres: the technological infrastructure, the services infrastructure and the policies infrastructure.

1. The computing technology infrastructure describes the technological progress, the software and hardware modules and the engineering processes to build these.

2. Having the technology in place is a prerequisite for creating new products and services which benefit from self-organizing computing. In their entirety, these new products and services build up the services infrastructure of potential SOC businesses.

3. Businesses, however, need rules, for protecting legitimate rights and properties of the participants.

The policies infrastructure describes a joint understanding and acceptance of rules, norms and laws as well as agreed-on measures to regulate and enforce compliance. The infrastructural spheres are not separate from each other. Without technology, there are no

services to be invented. Without services in action, no policies are needed to restrict or allow their usage. In a feedback loop, the requirement to express machine-readable rules to enforce software-based norms leads again to technological development. New concepts and technologies will inevitably again enable innovative services and at the same time require their regulation. Figure 2.1 shows the three infrastructural spheres schematically. Taking technological inventions into account as an existing, constant flow of fresh ideas and concepts of computer scientists, the research challenge will mainly lie in the realization of the services and the policies infrastructure. The following paragraphs describe selected research questions of these infrastructures in more detail.

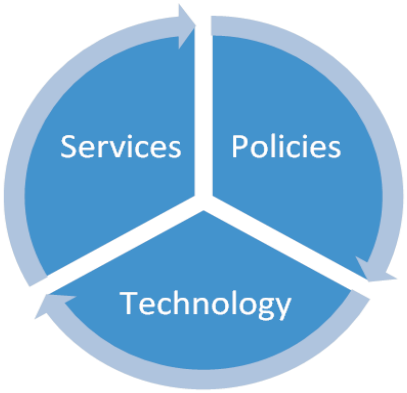

Figure 2.1: Infrastructural Spheres for Self-Organizing Computing

2.3 The Open Service Infrastructure

Most SOC environments are thought to be closed worlds, in which system elements provide resources benevolently. This is true for the usual scientific environments, e.g. Open Grid networks - the benefits for the users are counted in participating in scientific progress, and the promise of eventual reciprocity. In business environments, benevolent cooperation can be enforced - autonomous decision-making only allowed as far as it does not interfere with the business goals of the company. However, Autonomic Computing is not a concept to be confined within organizational borders. If we take visions of computing infrastructures serious, we will soon witness computing infrastructures comparable

to the super-national electricity grid (Grid Computing) or energy, water and other utilities grids (Utility Computing) [Car04]. These ubiquitous infrastructures are open and non-discriminating with regard to users wanting to join - anybody can participate in such an infrastructure. A small glimpse into the future might be the current trend 'Web 2.0', which recombines infrastructure software elements to create new services. Practically, demand and supply of new and existing services will define a services market. Markets have the advantage to collect existing resource supply and thus, usually, to achieve evenly utilization by leveling heterogeneous user behavior. Like other utilities, the services to be traded on those markets in huge numbers, are of simple nature. They are distinguishable by service quality characteristics, but convertible otherwise. This means that, given equal characteristics, competition will take place by signaling lower prices than the competition. However, the economic models underlying those markets are yet to be found. That includes the definition of the measure of utility, guiding the behavior of participants. Also methods which enable the transfer of goods and values (electronic money) have to be developed. Without such markets, SOC will stay yet another model for running cluster computing environments.

2.4 About the Necessity to Create an Open SOC Policies Infrastructure

For open 'utility computing' autonomic environments, we will soon encounter the demand for a policy infrastructure, like those in our non-electronic, physical society. Physical communities or networks frequently show very specific, only partly legally binding agreements and rules. Those policies define the resource-wise contributions of the network participants, the share of the utility gain, dealing with conflicts and mechanisms to enforce cooperative behavior of the participants. Self-organization requires all this, as the participants individually (autonomously) follow their pre-set goals. As computing systems always belong to humans (or businesses run by humans), human goals are those which ultimately define the course of the computational participants. SOC, by definition, is not a topic for synchronized, targeted and on-the-spot network management. It needs mechanisms to make risks calculable and process efficiently - without, the system as a whole will become incapable of action. For managing the individual, autonomous processes in the system one needs to find, enforce and monitor common rules, norms and institutions. Without those, we will face not intended consequences, non-innovation and resistance to application of that technology. Yet, the guarding policies are not fixed. With changing environment and changing system goals, even the policies need to adapt. This creates a meta-problem: the definition of policies for adaption of policies. Figure 2.2 shows a framework of elements to achieve spontaneous order. The circle begins with the software designers of the original system creating and defining an element's behavior. By way of a cultural evolution, rules of acceptable behavior get refined and give way to the next version of system inhabitants, who will be released in the information system and

shape it to their needs. The final open question is, whether the spontaneous order provides 'acceptable behavior' of the system - in principle, spontaneous order has no conscience and knows no guilt.

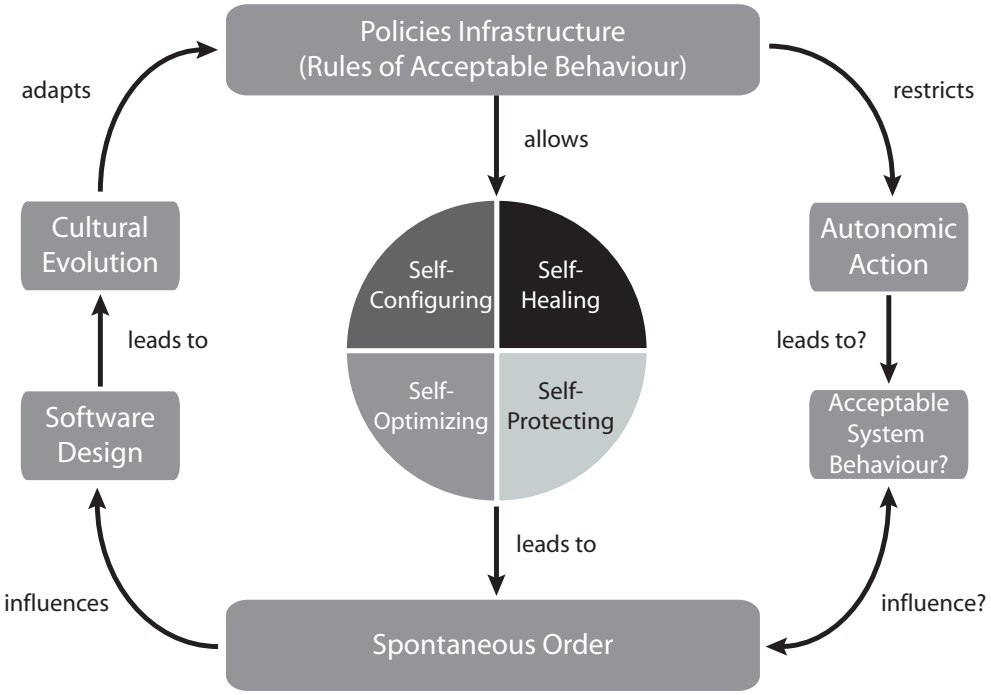

Figure 2.2: Regulatory Framework for Self-Organizing Computing

It is, however, possible to disrupt the self-coordination through targeted violation of the 'regulatory framework'. The automated pursuit of the individual goals alone does not produce an acceptable behavior of the entire system, e.g. in terms of robustness, controllability and the adherence of security criteria for the individual participant. Whether these perceptions can be generalized and used for the design of decentralized information systems (or information systems in a decentralized environment) and lead to more efficient paradigms for the implementation of computers, needs future research efforts. The question remains open, which the most effective framework to achieve spontaneous order is. The necessary regulation framework required can ex ante only be specified as trial and error. In this context, trust in, and generating trust through, Autonomic Computing are the main requirements for acceptability.

Chapter 3

Formal Description of Mechanisms

There are several competing approaches to how resource allocation is carried out. In this chapter, two of those are provided. The first mechanisms (Section 3.1) uses a centralized component in order to compute an allocation in ALNs. The second (Section 3.2) uses a decentralized approach in order to allocate demand and supply in such networks.

3.1 Centralized mechanisms

In the CATNETS scenario a two-tiered market architecture is considered (compare Figure 3.1).

As described and motivated in Y1 report [SNV+05b] in detail, we use a *double auction market* in order to carry out allocations in the service market. Due to the nature of the problem that has been analyzed in the first year of the project, a *multi-attribute combinatorial exchange* mechanism has been suggested for implementing the resource market. The mechanism is based on a centralized allocation between computational supply and demand. Supply and demand are expressed in terms of several attributes under consideration of different quality levels for each attribute.

The fundamental version of the mechanism is content of the Y1 deliverable [SNV+05b]. An extended version of the mechanism including robustness tests and standalone simulations has been published in [SNVW06].

The efforts concerning the centralized market mechanisms in CATNETS in Y2 have been focused on the implementation and integration into the simulator as well as the discussion and identification of measurable parameters to connect the metrics framework with the quantitative output of the simulation runs.

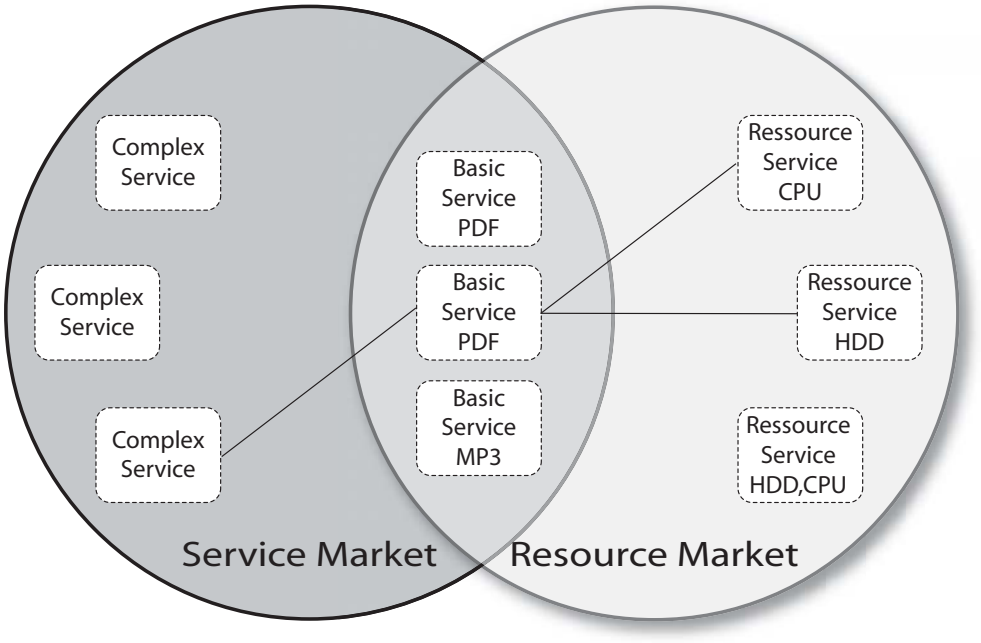

Figure 3.1: CATNETS Scenario: Service Market and Resource Market (Figure from Y1 Report)

3.2 The Catallaxy as an Alternative Decentralized Approach

Having defined a formal model for using a centralized economic allocation mechanism in a Grid network, this section describes an alternative, decentralized approach. The decentralized mechanism introduced here, implements the selection decision in the requesting client itself. Related realizations of decentralized approaches are found in P2P Networks, where Gnutella [AH00] is a typical example.

An optimization of network performance is out of the scope of the clients behavior; in contrast, the selfish conduct of each peer leads to performance and congestion problems in the P2P network, which are principally hard to solve [AH00]. Gnutella uses a flooding algorithm for service discovery (see Figure 3.2).

P2P algorithms are thus giving no guarantees whether a service is available at all. Furthermore their use of bandwidth is highly inefficient, as they were in 2004 responsible for approximately 56% of all network traffic measured by ISPs and 20% of the backbone traffic, counting only the search requests [AG04]. Innovative P2P approaches like Chord, Pastry or CAN avoid this message abundance by introducing an overlay structure [RFH$^+$01].

In decentralized matchmaking models, agents communicate directly with each other, decide on their own, and do not take the system state into account. In the Edgeworth

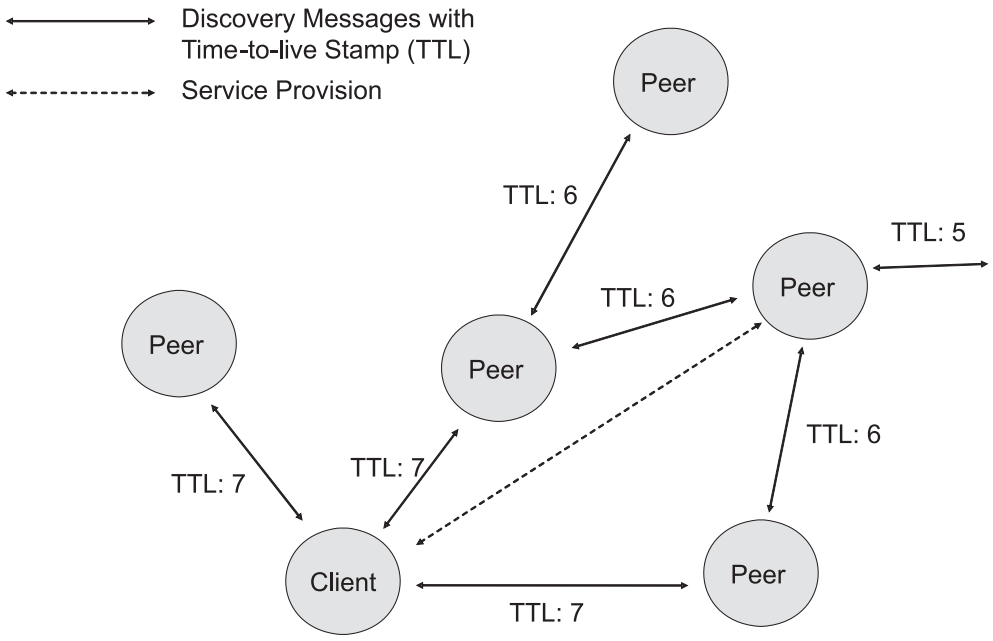

Figure 3.2: Decentralized Service Discovery

process [Var94], economic subjects trade bilaterally with each other only if their utility is supposed to increase after the barter. In that case, the sum of all utilities increases after each successful barter; the final state is Pareto-optimal and has maximum system utility.

A theoretical fundament how the concepts of dynamic market processes, heterogeneous agents and choice under incomplete information are linked, can be found in Neo-Austrian Economics, in particular in Friedrich August von Hayeks Catallaxy concept [HBKC89]. Catallaxy describes a state of spontaneous order, which comes into existence by the community members communicating (bartering) with each other and thus achieving a community goal that no single user has planned for.

The implementation of Catallaxy, described in this paper, uses efforts from both agent technology and economics, notably agent-based computational economics [Tes97]. Autonomous software agents negotiate with each other using an alternating offers protocol [Ros94] and adapt their negotiation strategies using feedback learning algorithms (evolutionary algorithms, numerical optimization e.g. Nelder/Meads simplex method [PT02], hybrid methods e.g. Brenners VID model [Bre02]). Ongoing communication by using price signaling leads to constant adaptation of the system as a whole and propagates changes in the scarcity of resources throughout the system. The resulting patterns are comparable to those witnessed in human market negotiation experiments [KR95][Pru81][Smi62].

3.2.1 Setup and Variables Definition

While the notation for buyers, sellers and goods is the same as the one used in the centralized case, we need to add definitions for the decision-making process (the strategy) of the agents. The negotiation strategy described here is based on the AVALANCHE strategy [ESP98][Eym01]. The strategy consists of 5 basic parameters, which define the individual behavior (genotype) of each agent.

For every tradable good there are two types of agents, buyers and sellers. Let agent k be a buyer and agent v a seller of a tradable good.

Let i_k be the number of negotiations that agent k has started and i_v the number of negotiations that agent v has started.

A genotype defines the behavior of the agents in the negotiation strategy. Let the genotype of agent $*$ for $* = k, v$ during his negotiation i_* be

$$G_*^{i_*} \in [0;1]^5$$

with

$$G_*^{i_*} = (G_{*,1}^{i_*}, \ldots, G_{*,5}^{i_*})^\tau = (a_*^{i_*}, s_*^{i_*}, t_*^{i_*}, b_*^{i_*}, w_*^{i_*})^\tau$$

where

$a_*^{i_*}$	acquisitiveness
$s_*^{i_*}$	satisfaction
$t_*^{i_*}$	priceStep
$b_*^{i_*}$	priceNext
$w_*^{i_*}$	weightMemory.

Acquisitiveness defines the probability of sticking with the last offer made, and not to make an unilateral concession in the following negotiation step. The value interval is between 0 and 1, and will be challenged by a stochastic probe in every negotiation step. A value of 0.7 means a probability of 70% that the agent will not make a concession – a highly competitive strategy. An agent with acquisitiveness value 1.0 will never change his price and an agent with *acquisitiveness* value 0.0 will always make an unilateral concession. If the probe succeeds, a buyer agent will rise his offer, a seller agent will lower his price.

The exact change of the bid value is defined by the concession level (*priceStep*). The concession level is represented by a percentage of the difference between the initial starting prices. A value of *priceStep* = 0.25 means a computation of the concession level as 1/4 of the first stated difference. If both opponents are homogenously negotiating and always concede, they meet each other on the half way in the third negotiation round under the assumption of no negotiation abortion.

Obviously, with an *acquisitiveness* level set high, and a *priceStep* set low enough, the opponents might never reach an agreement. The *satisfaction* parameter determines if an agent will drop out from an ongoing negotiation. The more steps the negotiation takes, or the more excessive the partner's offers are, the sooner the negotiation will be discontinued. Effectively, this parameter creates time pressure. Like for *acquisitiveness*, it does this by doing a stochastic probe against a set value between 0 and 1. A *satisfaction* value of 0.75 means, that the agent has a chance of 75% to continue the negotiation process. An agent with *satisfaction* = 0.0 will abort all negotiation at once and an agent with *satisfaction* = 1.0 will never abort.

The last piece of the strategy is an expression of selfishness. Behind each successful negotiation lies a future opportunity for gaining more of the utility share, by negotiating harder. *priceNext* thus modifies the starting bid. A successful seller will increase his offer price, a successful bidder will start with a lower bid next time.

For a viable strategy, the participants will have a close eye on what others deem to be the market price. If not, they risk being tagged as "excessive" and their bids will fail the *satisfaction* probe. They thus weigh current price information and historic price information in a specified ratio *weightMemory*, balancing short-time price fluctuation and longer-term opportunities.

At the beginning of the simulation the genes $G^{i_*}_{*,j}$ for $* = k, v$ and $j \in \{1, \ldots, 5\}$ are distributed according to the probabilities:

$$\text{Ufo}\big([m_j - \delta_j \,; m_j + \delta_j]\big)$$

Thereby, the constants m_j and δ_j for $j \in \{1, \ldots, 5\}$ are defined so that $[m_j - \delta_j \,; m_j + \delta_j] \subset [0; 1]$.

Additionally, each agent $*$ has the following variables:

$M^{i_*}_*$:	the market price, which is estimated by agent $*$ during his negotiation i_*.
$P^{i_*}_*$:	the price of the the last *successful* negotiation $1, 2, \ldots, i_*$ of agent $*$.
$O^{i_*}_*$:	the last offer, which the negotiation opponent has made in negotiation number i_* the agent $*$ before the negotiation ended.
$p^{i_*}_*$:	the number of stored plumages of agent $*$ direct after his negotiation i_*.

3.2.2 The Negotiation Strategy

When agent k and agent v negotiate, agent k is the buyer and agent v the seller. The sequence $(P_j)_{j \in \mathbb{N}_0} \subset [0, \infty[$ constitutes the offer in chronological order. The buyer always

makes the first offer. This means, all offers

$$P_{2m} \qquad \forall\, m \in \mathbb{N}_0$$

originate from the buyer and the offers

$$P_{2m+1} \qquad \forall\, m \in \mathbb{N}_0$$

come from the seller, where

$$m$$

is the negotiation round.

At the beginning of a negotiation the buyer k determines his initial price $\underline{K}$ and his maximum price $\overline{K}$:

$$\underline{K} = M_k^{i_k} \cdot (1 - b_k^{i_k})\,, \qquad \overline{K} = M_k^{i_k}$$

The seller v determines his starting price $\overline{V}$ and his minimum price $\underline{V}$:

$$\overline{V} = M_v^{i_v} \cdot (1 + b_v^{i_v})\,, \qquad \underline{V} = M_v^{i_v}$$

The buyer starts with the first bid:

$$P_0 = \underline{K}$$

- First Case: $\underline{K} \geq \overline{V}$
  Then v offers also
  $$P_1 = \underline{K}$$
  and the negotiation will be closed successfully to the price P_1.

- Second Case: $\underline{K} < \overline{V}$
  Then v offers his initial price
  $$P_1 = \overline{V}\ .$$
  Both agents now determine their steps δ_*^{j*} for price concessions:
  $$\delta_*^{i*} = (\overline{V} - \underline{K}) \cdot t_*^{i*} \qquad \text{for} \quad * = k, v$$

In the subsequent negotiation rounds, let $A_1, A_2, A_3, \ldots$ and $S_1, S_2, S_3, \ldots$ be stochastic independent random variables with the following binomial distributions:

$$A_{2m} = \begin{cases} 1 & \text{with probability} \quad a_k^{i_k} \\ 0 & \text{with probability} \quad 1 - a_k^{i_k} \end{cases} \qquad \forall\, m \in \mathbb{N}$$

$$A_{2m+1} = \begin{cases} 1 & \text{with probability} \quad a_v^{i_v} \\ 0 & \text{with probability} \quad 1 - a_v^{i_v} \end{cases} \qquad \forall\, m \in \mathbb{N}$$

$$S_{2m} = \begin{cases} 1 & \text{with probability} \quad s_k^{i_k} \\ 0 & \text{with probability} \quad 1 - s_k^{i_k} \end{cases} \qquad \forall\, m \in \mathbb{N}$$

$$S_{2m+1} = \begin{cases} 1 & \text{with probability} \quad s_v^{i_v} \\ 0 & \text{with probability} \quad 1 - s_v^{i_v} \end{cases} \qquad \forall\, m \in \mathbb{N}$$

- Offer number $2m$; it is the buyer's k turn:

  If $S_{2m} = 0$ and $P_{2m-1} \geq P_{2(m-1)-1}$ with $m \neq 1$, then the buyer k cancels the negotiation. This means, $O_k^{i_k} = P_{2m-1}$ and $O_v^{i_v} = P_{2(m-1)}$.

  Otherwise, the buyer k makes the following offer:

$$P_{2m} = \left(\min \left\{ \overline{K}, (P_{2(m-1)} + \delta_k^{i_k}), P_{2m-1} \right\} \right)^{1-A_{2m}} \cdot$$

$$\left(P_{2(m-1)} \right)^{A_{2m}}$$

- Bid number $2m + 1$; it is the seller's v turn:

  If $S_{2m+1} = 0$ and $P_{2m} \leq P_{2(m-1)}$, then the seller v cancels the negotiation. That means, $O_v^{i_v} = P_{2m}$ and $O_k^{i_k} = P_{2(m-1)+1}$.

  Otherwise the seller v makes the following offer:

$$P_{2m+1} = \left(\min \left\{ \underline{V},\ (P_{2(m-1)+1} - \delta_v^{i_v}),\ P_{2m} \right\} \right)^{A_{2m+1}} \cdot$$

$$\left(P_{2(m-1)} \right)^{1-A_{2m+1}}$$

The negotiation ends if either one of the agents cancels the negotiation or the negotiation ends successfully with

$$P_j = P_{j+1}$$

for a $j \in \mathbb{N}$. In this case, it holds $O_k^{i_k} = P_j = O_v^{i_v}$.

With the end of a *successful* negotiation to the price P_j the negotiation compute their estimated *profit*

$$\Pi_k^{i_k} = M_k^{i_k} - P_j \qquad \text{respectively} \qquad \Pi_v^{i_v} = P_j - M_v^{i_v} \ .$$

Additionally, both agents update after *every* negotiation their estimated market price using

$$M_k^{i_k+1} = w_k^{i_k} \cdot O_k^{i_k} + (1 - w_k^{i_k}) \cdot M_k^{i_k}$$

respectively

$$M_v^{i_v+1} = w_v^{i_v} \cdot O_v^{i_v} + (1 - w_v^{i_v}) \cdot M_v^{i_v} \ .$$

This last step is independent of the success of a negotiation.

3.2.3 Gossip Learning

The learning concept used in this simulation is derived from so-called gossip learning. This means that the agents learn from received information about other transactions in the market. This information may not be accurate or complete, but serves as an indication about the gross direction of the market. In our implementation, this gossip information is created and broadcast by a successful agent, in analogy to issuing an ad-hoc information in stock market periodicals.

Let n be an agent and $g_1, \ldots, g_d$ the tradable goods. The agent n has finished his negotiation i_n successfully with an estimated profit of $\Pi_n^{i_n}(g)$ for the good $g \in \{g_1, \ldots, g_d\}$. A *learning* step according to the learning algorithm (see subsection 3.2.4) is performed by agent n last time at the end of his negotiation j_k. This means

$$G_n^{j_n+1} = G_n^{j_n+2} = \cdots = G_n^{i_n}.$$

If agent n with the negotiation numbers

$$j_n + 1, j_n + 2, \ldots, i_n$$

has successfully completed at least 10 negotiations for every good, he sends a *Plumage*

$$(G_n^{i_n}, F_n^{i_n})$$

to all other agents of his type. Then, his updated *fitness* is $F_n^{i_n}$, which is computed as follows:

(a) For every good $g_j \in \{g_1, \ldots, g_d\}$ the next profit value $\Pi(g_j)$ is determined: Let

$$\Pi_1(g_j), \ldots, \Pi_{10}(g_j)$$

be the estimated profits of the last 10 successful negotiations of agent n for the good g_j. Then, the fitness is

$$F_n^{i_n}(g_j) = \frac{1}{10}\Big(\Pi_1(g_j) + \cdots + \Pi_{10}(g_j)\Big) .$$

(b) The updated fitness $F_n^{i_n}$ finally is

$$F_n^{i_n} = \frac{1}{d}\Big(\Pi(g_1) + \cdots + \Pi(g_d)\Big) .$$

3.2.4 The Learning Algorithm

It is assumed that the agents show a cooperative behavior. This means, the agents report truthfully their learning information.

After having received some gossip information message, the agent may modify his own strategy. The comparison of the own results with those of the strategy received may show, that the other strategy superior to the own. In this case, the agent will try to cross both strategies to gain competitive advantage. In practice, out of a list of received genotype/performance-tuples, the agent will choose the best performing external genotype, and then mix, cross and mutate with his own genotype.

Let be n an arbitrary agent at the end of his negotiation i_n and let be $p_n^{i_n}$ the number of plumages, the agent n has stored directly after his negotiation i_n. The last *learning* step was performed by agent n after his negotiation j_k. Let be $e_n^{i_n}$ the number of negotiations, an agent n of the negotiation numbers

$$j_n + 1, j_n + 2, \ldots, i_n$$

has successfully finished.

Let be

$$p = 1 \ .$$

If

$$p_n^{i_n} < p \qquad \text{or} \qquad e_n^{i_n} < 10$$

applies for agent n after his negotiation i_n, no *learning* step will be performed. This means, his genotype will not change:

$$G_n^{i_n+1} = G_n^{i_n} \ .$$

Hence, if

$$p_n^{i_n} \geq p \qquad \text{and} \qquad e_n^{i_n} \geq 10$$

applies, the agent n performs a *learning* step. The genotype of agent n changes as follows:

First, the stored plumage of agent n with the highest fitness is selected. Let be

$$G_f = (G_{f,1}, \ldots, G_{f,5})^\tau = (a_f, s_f, t_f, b_f, w_f)^\tau$$

the related genotype. Second, a *crossover* is performed. In doing so, a new genotype $\tilde{G}_n^{i_n+1}$ is created, which contains a random mixture of genes of the genotypes $G_n^{i_n}$ and G_f. This process follows a *mutation* step third: Using the genotype $\tilde{G}_n^{i_n+1}$ and changing its genes slightly will result in the genotype $G_n^{i_{n+1}}$.

3.2.4.1 Crossover

Let be $C_1, \ldots, C_5$ stochastic independent random variables with the following binomial distribution:

$$C_j \ = \ \begin{cases} 1 & \text{with probability} \quad 0,5 \\ 0 & \text{with probability} \quad 0,5 \end{cases} \qquad \forall\, j \in \{1, \ldots, 5\}$$

Then it is imperative

$$\tilde{G}_{n,j}^{i_n+1} = (1 - C_j) \cdot G_{n,j}^{i_n} + C_j \cdot G_{f,j} \qquad \forall\, j \in \{1, \ldots, 5\}\ .$$

3.2.4.2 Mutation

Let be $M_1, \ldots, M_5, X_1, \ldots, X_5$ stochastic independent random variables with the following distributions:

$$\begin{aligned} M_j &= \begin{cases} 1 & \text{with probability} \quad 0,05 \\ 0 & \text{with probability} \quad 0,05 \end{cases} \quad \forall\, j \in \{1, \ldots, 5\} \\ X_j &\sim \mathcal{N}(0\,,\,1) \qquad \forall\, j \in \{1, \ldots, 5\} \end{aligned}$$

That means, X_j is $\forall\, j \in \{1, \ldots, 5\}$ standard normal distributed.
Then, it holds

$$\begin{aligned} G_{n,j}^{i_n+1} = \max\Big\{0\,;\, \min\big\{\tilde{G}_{n,j}^{i_n+1} + \\ M_j \cdot \big((\frac{1}{10} X_j) \bmod(1)\big)\,;\, 1\big\}\Big\} \qquad \forall\, j \in \{1, \ldots, 5\}\ . \end{aligned}$$

Chapter 4

Bidding Issues

The purpose of this chapter is to clarify what and when an agent bids in the CATNETS scenario. *What* denotes the valuation and the reservation prices of agents, i.e. the maximal price which an agent is willing to pay for a service (resp. the minimum price an agent has for selling a service). *When* denotes the timing of bids, i.e. which event induces an agent to bid for a service. Both cases are different in the centralized and decentralized scenario. As such, it is important to find concepts that are applicable for both scenarios and, thus, make the results comparable.

The chapter is structured as follows: Section 4.1 outlines the valuation generation of an agent, i.e. the procedure that determines the value of an agent's bid. Section 4.2 describes the timing of the agent's bids, i.e. when does an agent bid for a service. Finally, Section 4.3 summarizes the chapter.

4.1 What Does an Agent Bid?

The following section describes what an agent bids, i.e. the valuation and reservation prices. For this, a generic function is developed that is applicable for both, the centralized and the decentralized case. The concept is applied for buyers and sellers in both markets, i.e. in the service market and in the resource market.

4.1.1 Notation

Before the valuation generator is introduced, the general notation as denoted in table 4.1 is presented.

The transaction object g denotes the service for that a valuation is to be generated.

Transaction object	g
Valuation in period i	V^i
Market price	M^i
Weight market price	β
Weighted Average	wav^i
Weighted Memory	w^i

Table 4.1: Notation for the valuation generation

For instance, this could be a PDF creator in the service market. For each service, an agent has a valuation V^i (resp. reservation price) in each period i. This valuation denotes the maximum price, an agent is willing to bid for this particular service.

The valuation generation is influenced by external factors such as the market price. In case such a market price exists for a transaction object in period i, it is denoted by M^i. For the centralized case, market prices may not exists in each period. In these cases, an approximated market price is used. The effect a market price has on the valuation generation is denoted by β. The lower this value is, the lesser the importance of old market prices.

Finally, the weighted average wav^i and the weighted memory w^i denote noise parameters.

4.1.2 Valuation Generation

Based upon the notation as introduced in the previous section, the valuation for a service g in time period $i+1$ is calculated as follows:

$$V^{i+1}(g) = \beta M^i(g) + (1-\beta)wav^i(g) + \mathcal{Y}\mathcal{X} + \mathcal{Z} \tag{4.1}$$

The valuation for the next period depends on the market price of the current period, the weighted average of former valuations, and some statistical noise. The weight of the market price and the weighted average depends on the static value $\beta \in \{0, 1\}$ which is predefined and fixed. From an implementation point-of-view, the variable β should be definable via an external configuration file.

It is to note, that the statistical noise functions are only applied in the centralized case. These functions are responsible for inserting exogenous factors (e.g. different dynamics, density scenarios) in the centralized simulation. In the decentralized case, this noise is generated by a genetic algorithm. However, this genetic algorithm will not be

applied to the centralized auctioneer, as it is a special algorithm for the decentralized case: The algorithm (decentralized learning strategy) serves for information propagation in CATNETS. However, we do not need this propagation in the centralized case, because the auctioneer spreads all relevant information out of a central point. The challenge is, however, to define noise functions that generate "similar" values as in the decentralized case.

The calculation of the weighted average and the statistical noise are discussed in the following.

4.1.2.1 Calculating Weighted Average

The weighted average for a service g in period i is calculated as follows:

$$
\begin{aligned}
wav^{i}(g) &= w^{i}V^{i} + (1 - w^{i})wav^{i-1}(g) && (4.2)\\
wav^{i-1}(g) &= w^{i-1}V^{i-1} + (1 - w^{i-1})wav^{i-2}(g) && (4.3)\\
\cdots &= \cdots && (4.4)\\
wav^{i-j}(g) &= w^{i-j}V^{i-j} + (1 - w^{i-j})wav^{i-j-1}(g) && (4.5)
\end{aligned}
$$

The value will be re-calculated after each transaction session, i.e. after each round.

The function is recursive and its depth is limited by the factor j. The required old valuations V^i are stored in a ring-array with j places, where j has to be definable via an external configuration file. The difference between the centralized and decentralized case is, that j may be different in both cases. This is reasoned by the fact, that more valuations are generated in the decentralized case than in the centralized case. For instance, in the centralized case only one valuation is generated per transaction session (one-shot auction). In the decentralized case, however, multiple valuations (at most 3) may be generated during one transaction session. In order to respect this difference, j should be smaller in the centralized case than in the decentralized one. For instance, this value could be $j_{\text{centralized}} = j_{\text{decentralized}}/10$.

4.1.2.2 Statistical Noise

Statistical noise is required to add externalities to the valuation generation. In practice, these externalities could be some information or news distributed to the agent. For instance, this could be an external information that a large computer center is out of order for several weeks. Furthermore, this noise is required to avoid "deadlocks" of the simulation, i.e. situations in which all prices converge to a specific point.

For the CATNETS valuation generation, the following noise functions are applied: $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$.

$$\mathcal{Y} = \begin{cases} 1 & \text{with probability of } p = 0.5 \\ 0 & \text{with probability of } p = 0.5 \end{cases} \tag{4.6}$$

$\mathcal{Y}$ is a simple binary variable which determines whether the value computed by $\mathcal{X}$ will be used or not. It is drawn from a uniform distribution. The value $\mathcal{Y}$ in the centralized case is comparable to the crossover operator in the decentralized case.

The value $\mathcal{X}$ is drawn from a normal distribution, i.e.

$$\mathcal{X} \sim \mathcal{N}(a, b), \tag{4.7}$$

where a is the mean of the distribution and b is the standard deviation. In addition, a scaling factor will be required, if the distribution is $\mathcal{X} \sim \mathcal{N}(0, 1)$. $\mathcal{X}$ in the centralized case is comparable to the *priceNext* operator in the decentralized case.

Finally, the value $\mathcal{Z}$ is also drawn from a normal distribution, i.e.

$$\mathcal{Z} \sim \mathcal{N}(c, d), \tag{4.8}$$

where c is the mean and d its standard deviation. Similar to $\mathcal{X}$, we may apply a scaling factor. $\mathcal{Z}$ in the centralized case is comparable to the mutation operator in the decentralized case. It has to be small enough to avoid high price jumps (see decentralized case with p=0.05).

4.2 When Does an Agent Bid?

The value of an agent's bid can be calculated by means of the defined valuation generator. In the next step, we have to define the timing of the agent's bid, i.e. we have to define events when an agent bids in the centralized and decentralized case.

In the following, the timing concepts are outlined for each type of market participant: for the Complex Service Agent, the Basic Service Agent, and the Resource Service Agent. Furthermore, it is distinguished between the centralized and the decentralized case.

4.2.1 Complex Service Agent

The Complex Service Agent bids on the service market, whenever it receives a request from an application. This request is triggered and, as a result, exogenously given. The actions of the agent in the centralized and decentralized case are as follows:

Centralized: The agent submits a bid to the auctioneer.

Decentralized: The agent starts to distribute a call-for-proposal (CFP).

4.2.2 Basic Service Agent

For the basic service agent, we identified three alternatives (strategies) that can be applied to the timing of bids. It is to note, that the strategies encompass the basic service agent's role of a seller in the service market and that of a buyer in the resource market.

In the following, these strategies and their applicability for CATNETS are discussed.

4.2.2.1 Strategy A

The first strategy for the basic service agent is defined as follows (see Figure 4.1):

Centralized: In case a basic service has free capacity[1], it starts to get some new resources on the resource market (1). For this, it bids on the resource market until all required resources for the basic service are purchased (2). Afterwards, it submits an offer to the service market (3).

Decentralized: In case a basic service has free capacity, it starts to get some new resources on the resource market (1). For this, it bids on the resource market (i.e., it distributes CFP's) until all required resources are purchased (2). After that, it waits until it gets a CFP on the service market (3).

The problem of this strategy is, that all required resources on the resource market have to be locked until an agreement on the service market is reached (4). In the current CATNETS system, this is not realizable, as no time attributes are included.

4.2.2.2 Strategy B

The second strategy for the basic service agent is defined as follows (see Figure 4.2):

[1] This value will be measured by the simulator.

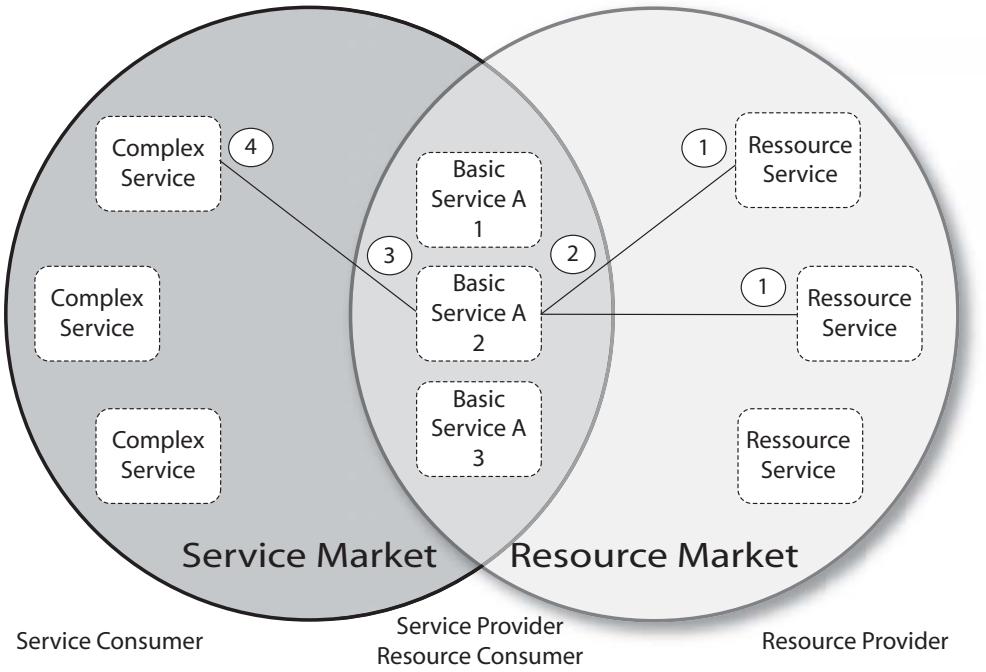

Figure 4.1: Bidding Strategy A

Centralized: The basic service agent bids, whenever it has free capacity on the service market (1). If an agreement is reached (2), it tries to get the resources on the resource market (3)(4). For this, it has only one shot, i.e. one bid. If something fails on the resource market, the agreement on the service market will be rejected.

Decentralized: The basic service agent replies to a CFP on the service market (1)(2), whenever it has free capacity. After that, it starts negotiations on the resource market (3)(4). If something fails on the resource market, the agreement on the service market will be rejected.

This strategy is applicable for the centralized and decentralized case.

4.2.2.3 Strategy C

The third strategy for the basic service agent is defined as follows (see Figure 4.3):

Centralized: The basic service agent takes a look into the order book of the service market (1). In case, a request is submitted to the order book, the basic service agent tries to get resources for that on the resource market (2)(3). If resources are traded, it submits a corresponding order to the service market (4).

Decentralized: If the agent receives a CFP on the service market (1), it replies only if it has corresponding agreements on the resource market (2)(3). After that, it continues

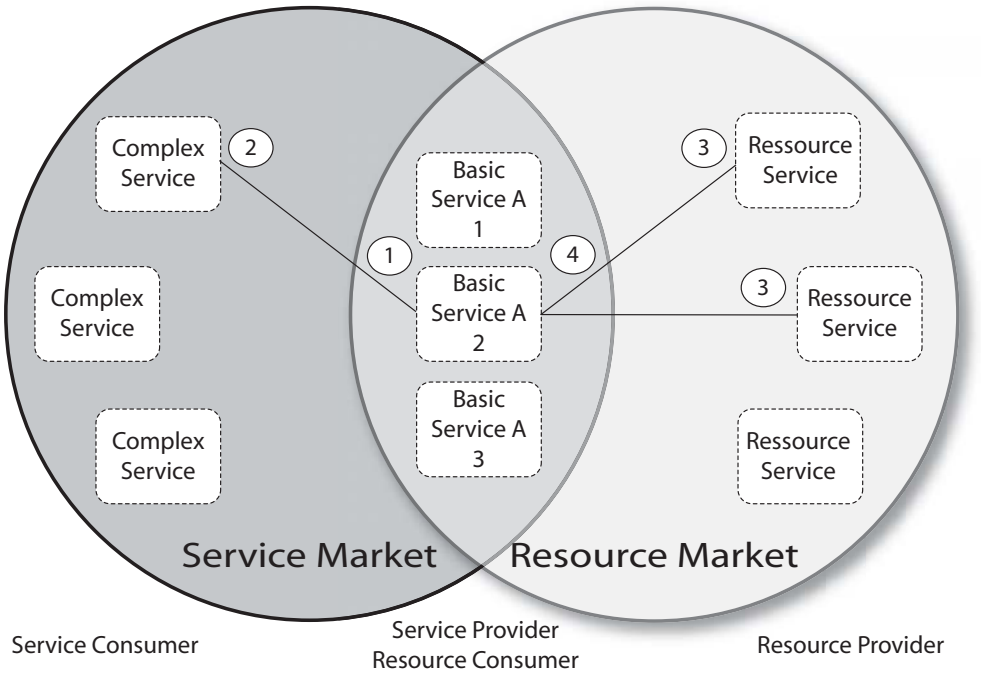

Figure 4.2: Bidding Strategy B

the negotiations on the service market (4), although the agent is not sure if it gets selected (is successful) on the service market.

The problem of this strategy is the very high probability of rejecting a lot of agreements. Furthermore, this alternative is hard to realize in the centralized case.

4.2.2.4 Summary

Due to the aforementioned drawbacks of strategy A and strategy C, we selected strategy B for CATNETS.

4.2.3 Resource Service Agent

Finally, for the resource service agent, the strategy is defined as follows:

Centralized: The resource service agent bids whenever it has free capacity. It submits a bundle order containing all its available (and free) resources. But the auctioneer is able to split bundles if that raises the overall welfare.

Decentralized: The resource service agent replies to a CFP, if it has free capacity. It offers its maximum free capacity (replies a message containing all resources contained in the bundle, that are available at the resource agent's site). For instance,

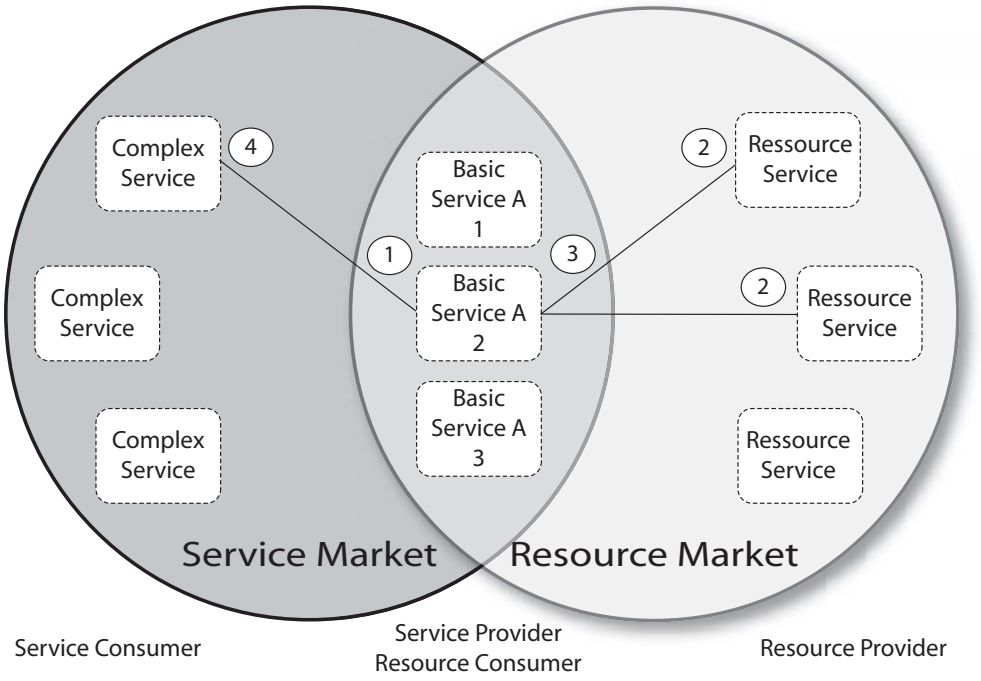

Figure 4.3: Bidding Strategy C

having a request for the bundle $\{A, B, C\}$ and a resource agent with the available resources $\{A, C, D\}$, the agent would reply $\{A, C\}$ to the CFP. From an implementation point-of-view, this will be realized by means of buckets for each resource combination. It is assumed that the value of a resource bundle is higher than the sum of values of the single resources it is composed of. So the valuation of the bundle $\{A, B, C\}$ would be higher than the sum of valuations of the resources $\{A\}$,$\{B\}$ and $\{C\}$.

4.3 Summary

This section outlines bidding issues in the CATNETS scenario. A valuation generator is introduced that can be applied for buyers and sellers in both markets in order to determine values for their bids. The challenge of such a generator is to define a concept that is applicable for the centralized and the decentralized case and that leads to comparable outcomes.

Chapter 5

Integration of Mechanisms into Simulator

Subject to this chapter is the technical integration of the centralized and decentralized mechanisms into the simulator. Here, a string interaction with WP2 will be carried out in order to avoid overlaps in documentations.

5.1 Integration of the Auction Mechanisms

The objective of this section is to describe the implementation of the auction mechanisms (Section 5.1.1), their integration into OptorSim (Section 5.1.2), and to give a brief overview of the results obtained from preliminary simulation runs (Section 5.1.3).

5.1.1 Implementation of the Markets

In the following, the implementation of the service market and the resource market is described. Both market mechanisms are implemented as independent software services which allows us to integrate them into other systems easily. Beside the integration into OptorSim, this flexibility allows us to integrate the markets into the prototype in the future such as proposed in [CJSF06].

5.1.1.1 Service Market

As outlined in the last deliverable [SNV+05a], a double auction is applied to the service market. In a double auction market [Fri91], a large number of participants trade a common object and can submit bids (buy orders) and asks (sell orders). Trading in double auctions is organized by means of order books, each for a set of homogeneous

goods. In the CATNETS scenario there will be n different order books, each for one of the n different services.

Figure 5.1 depicts the high level architecture of the service market for CATNETS [SNV+05a]. Complex service agents can submit buy orders to the order books; basic service agents can submit sell orders. Each set of homogeneous services (e.g. PDF creator services) is traded in a single order book.

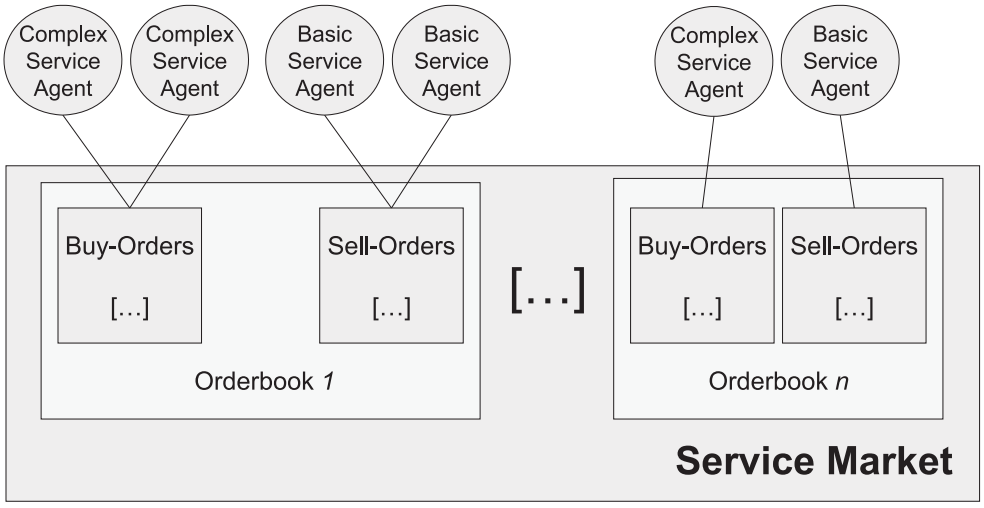

Figure 5.1: The service market including several double auction order books.

A simplified class diagram of the service market implementation is shown in figure 5.2: For each type of basic service traded in the service market, an instance of the `Orderbook` class is generated. The order book provides functionality to add orders, remove them, and to start the outcome determination. Whenever an agent wants to submit an order to the market, it generates an instance of the `Order` class and submits it to the order book. The order book is also responsible for triggering the clearing process. In case, a continuous clearing is used, the order book instantiates the `Allocator_CDA` class; otherwise it uses the call market as implemented in the `Allocator_CallMarket` class. Both allocator classes use the matchmaker `Match` to find corresponding counterpart orders. After the allocation and the prices are computed, an `Allocation` object is generated for each transaction. This object points to the parties that are involved in the transaction, i.e. it points to an order from a buyer and an order from a seller. The allocation objects are stored in a vector and can be parsed by the simulator.

As the diagram shows, the implementation supports continuous clearing and a call market. In a continuous clearing auction, buyers and sellers simultaneously and asynchronously announce bids and offers. In case a new order enters the market, the auctioneer tries to clear the market immediately. A call market is an auction with periodic uniform clearing, e.g. the auctioneer clears the market every fives minutes. All orders

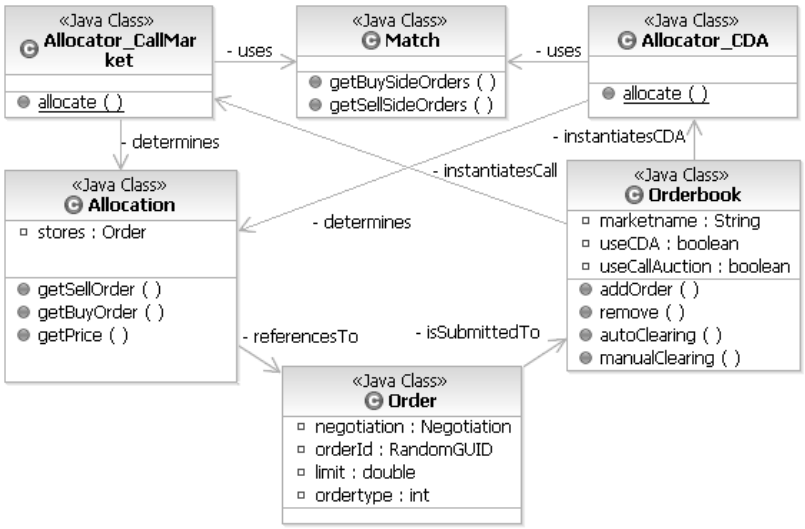

Figure 5.2: Class diagram of the most fundamental classes of the service market

in a period are collected in an order book and will be cleared periodically [SNV+05a]. The clearing strategy that is applied can be selected by means of a configuration file. The evaluation runs in the next project year will give us insights, which clearing strategy is superior.

5.1.1.2 Resource Market

For allocating services in the resource market, we apply a multi-attribute combinatorial exchange (MACE) [SNV+05a, SNVW06]. Figure 5.3 depicts the sequence of the auction in the CATNETS scenario. Agents (buyers and sellers) submit their bids to the auctioneer instance. After that, the bids are transformed into an internal representation form and, subsequently, the winners are computed (allocation). Finally, prices are computed in consideration of the allocation. As a result of the market mechanism, the agents get informed whether or not they are part of the allocation.

Figure 5.4 depicts some of the most basic components of the implementation as a UML class diagram: The `Market` class is the central component of the implementation. On the one hand, it is responsible for initializing all relevant classes. On the other hand, the market class controls the process to submit orders and to compute an outcome. `Agents` can submit orders to an order book, where an order consists of a price and a `Bundle`, where a bundle is a collection of `Good` instances. After the bids are submitted by the agents, an `Outcome` is computed. For this, the market uses a `ModelFactory` and a `PricingFactory`. The `ModelFactory` is responsible for providing a winner determination model in order to compute an allocation. In the CATNETS

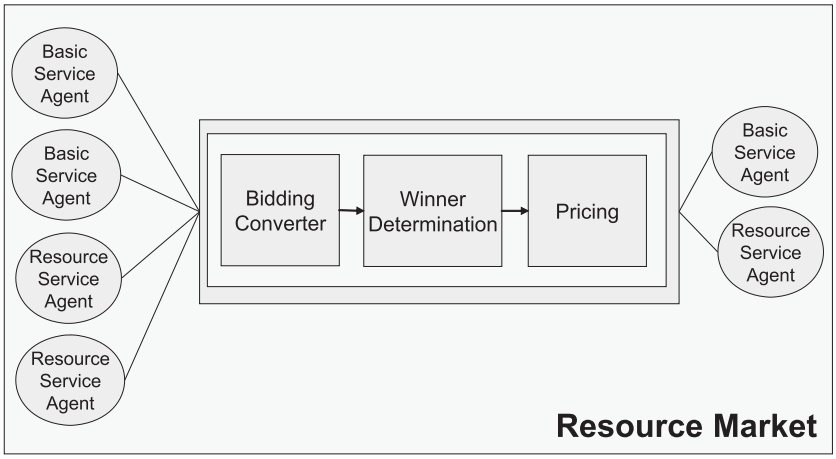

Figure 5.3: Sequence of an auction for the resource market

scenario, this model is implemented in the `CatnetsResourceMarket` class. The `PricingFactory` provides a set of price mechanisms that can be used. In CATNETS, the pricing schema as implemented in the `KPrice` class is applied. After an allocation and corresponding prices are determined, the result is stored in the `Outcome` object. This object can be queried in order to retrieve the required information such as allocation decisions and prices for each transaction.

For implementing the resource market, we make use of the standard linear programming libraries CPLEX and LPSolve. CPLEX is a commercial product and is currently the state of the art optimization engine[1]. LPSolve is a free linear programming solver that implements the branch-and-bound method for solving integer problems[2]. CPLEX is currently one of the fastest solving libraries and will be used for the evaluation of the mechanisms. The use of LPSolve (more specifically, its license) allows us to install the resource market implementation on every machine. As such, the development and testing of the mechanisms can be fostered.

The behavior of the implementation can be controlled by means of configuration files. Among others, several alternative pricing schemas are implemented, e.g. the approximated Vickrey pricing algorithm [PKE01]. The concrete pricing mechanism can be selected by means of a configuration parameter. Furthermore, the clearing interval[3] of the market can be controlled by the configuration file.

[1] See `http://www.cplex.com/` for details.
[2] See `http://www.geocities.com/lpsolve/` for details.
[3] The resource market is currently restricted to a periodical clearing.

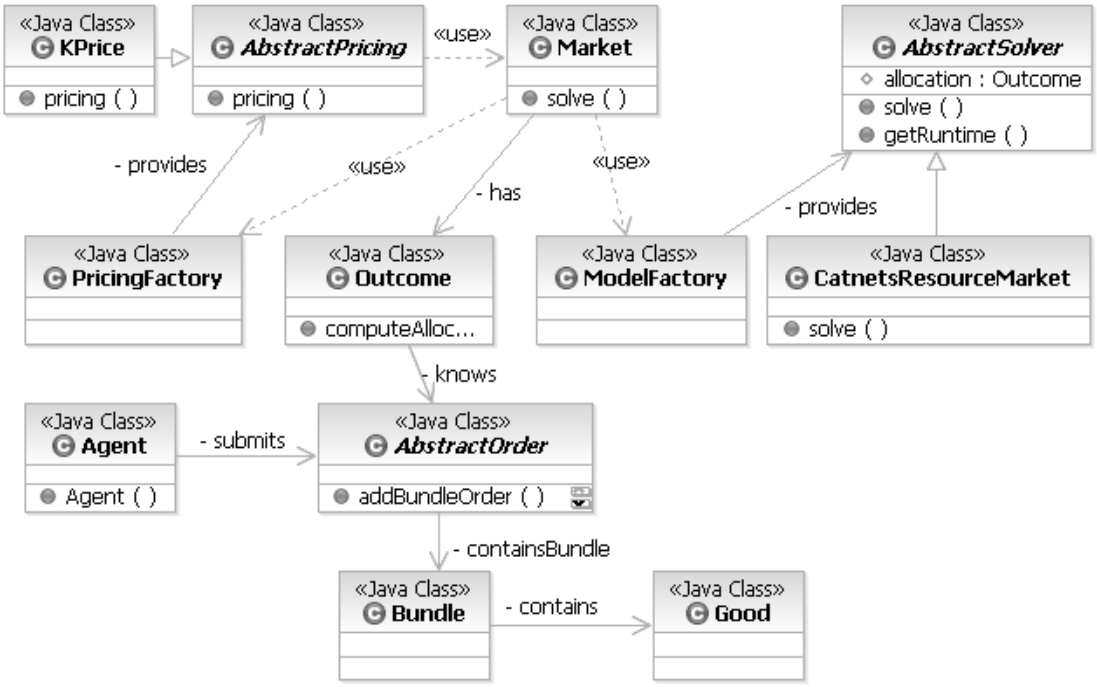

Figure 5.4: Class diagram of the most fundamental classes of the resource market

5.1.2 Integration into OptorSim

This section briefly outlines the integration of the markets into OptorSim. For a detailed description of how these concepts are implemented, the reader is referred to deliverable D2.2 [CSSZ06].

In OptorSim, the auctioneers for the service market and resource market are both realized as agents. They get instantiated by the simulator during its initialization and can be contacted by every other agent. Agents communicate with the auctioneers in order to submit their bids and to retrieve status information such as the last market price for a service or the allocation decision. The communication between trading agents and auctioneers is realized by means of messages.

Figure 5.5 shows the general interaction between trading agents and an auctioneer. Each time, a complex service agent (CSA) wants to acquire of service, it submits a message to its Peer-to-Peer Manager (P2P). This manager is capable of advertising and routing messages to other agents. In case a manager receives a message from its CSA, it forwards the message to the auctioneer (SMAA). Likewise, a basic service agent (BSA) can also submit an offer message to its Peer-to-Peer Manager which also forwards it to the auctioneer. On the basis of the messages, the auctioneer computes an outcome, i.e.

allocation decisions and prices. The result of this outcome (successful or unsuccessful bid) is subsequently sent back to the agents. The figure shows the process for the service market exemplarily; for the resource market the process is identical.

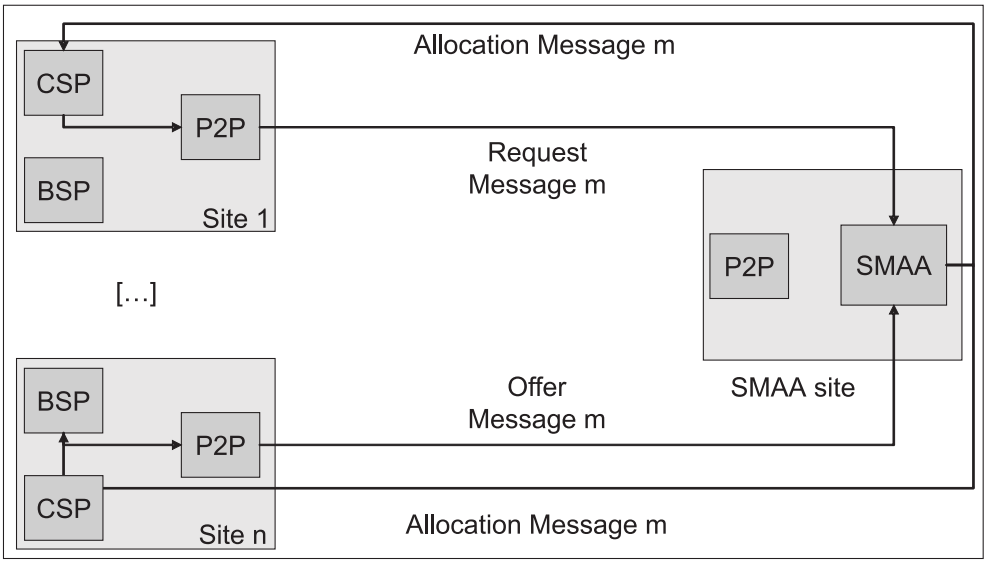

Figure 5.5: Sequence of an auction for the resource market.

The central interfaces between OptorSim and the auctioneers are messages. From a conceptual point of view, different message types are required for different actions of the agents:

Request Message: A request message is sent, whenever an agent wants to buy a service. For instance, a complex service agent submits such a message to the auctioneer to bid for a basic service. The message contains information about the transaction object (which service), the agent's valuation price (maximum price), as well as an ID of the agent.

Offer Message: An offer message is sent, whenever an agent wants to sell a service. For instance, a resource service agent submits such a message when it wants to sell its resources. In analogy to the request message, the offer message contains information about the transaction object (which service), the agent's reservation price (minimum price), as as well as an ID of the agent.

Allocation Message: After the auctioneer has computed an outcome, allocation messages are sent back to each participating agent. In case the agent was successful in the auction (i.e. it is part of the allocation), the message contains information about the price of the service and its counterpart. For instance, if a basic service agent retrieves a successful allocation message from the resource market auctioneer, the message contains information about the resource service agent who will

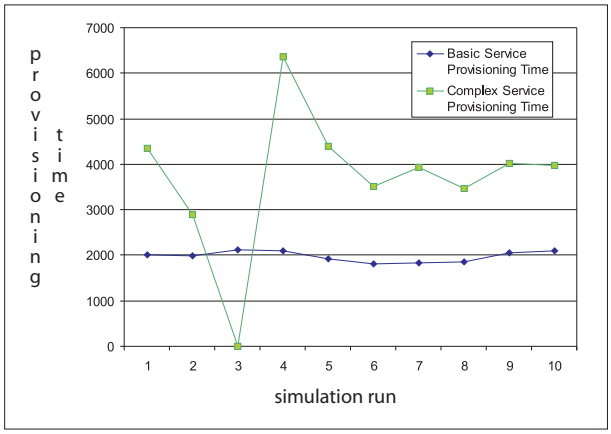

Figure 5.6: Basic service and Complex service provisioning time

provide the resources. In case an agent was unsuccessful, the contains a negative price ($p = -1$).

Delete Message: Sometimes agents need to cancel orders which they have submitted to the auctioneer. In this case, they submit a delete message to auctioneer. This message contains all relevant information such as the agent ID and an order ID.

Each message type is implemented for the service market and the resource market. The implementation of the messages is described in the deliverable D2.2 in more detail [CSSZ06].

5.1.3 Results from Refinement Runs/Simulations

For a proof-of-concept evaluation of the simulator, we have run several simulation runs to test the integration of the auctions. In this section, we will provide some preliminary results from these tests. The results are neither meant to be convincing nor to be statistically evident. This work will be done in project year 3.

We have run 10 different simulation runs using a small configuration file with 3 agents and 2 different resources (cf. WP 2). We measured a subset of the metrics defined in WP4.

Figure 5.6 depicts the average *basic service* and *complex service provisioning time* for each simulation run. While the basic service provisioning time is approximatively constant, the complex service provisioning time fluctuates in run 3. In this run, no complex service was successfully allocated by the mechanism, i.e., in this run, the reservation prices of the sellers are greater or equal to the buyers' valuations.

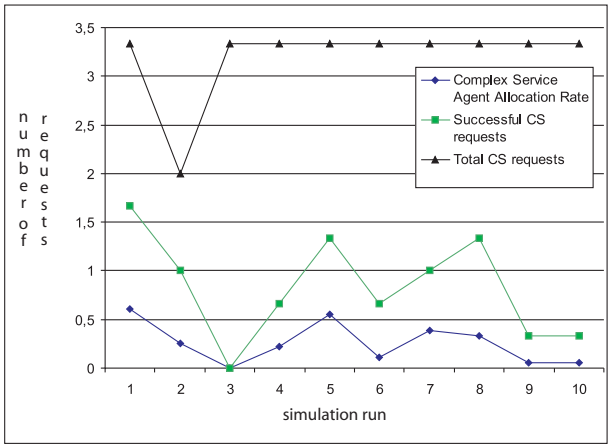

Figure 5.7: Allocation rates and Complex service requests

Figure 5.7 depicts the *complex service (CS) agent allocation rate*, the number of *successful CS requests*, and the number of *total CS requests*. It is obvious that the total number of CS requests is greater than the number of allocated ones. In order to improve this ratio, we have to balance the valuation generation of the agents (cf. Chapter 4) as well as the strategies of the bidding agents in project year 3.

5.2 Decentralized Mechanisms (Catallaxy)

This section briefly outlines the implementation of the decentralized markets and its integration into OptorSim. Section 5.2.1 focuses on the implementation of the decentralized (catallactic) service and resource market. The implementation of the strategy module and its adoption to the markets are described in detail. The integration into OptorSim, message patterns and the introduced message types describes Section 5.2.2. Preliminary results from simulation runs are presented in Section 5.2.3.

5.2.1 Implementation of the Markets

The implementation of the service and resource markets use the catallactic reasoner implementation presented in the deliverable D1.1. Both markets use the same strategy implementation for reasoning about proposals. Therefore, the objective of this section is to describe the implementation of the service and resource market using the catallactic reasoner.

Differences between the service and the resource market occur at the initialization of the strategy, their interaction pattern and the integration into the simulator and prototype environment. In the following, the focus lies on the interfaces of the strategy and its initialization. The interaction patterns are dependent on the environment to be integrated.

Thus, they are described in their corresponding sections.

The interface of the reasoner shows figure 5.8. The implementation offers customized reasoners for every agent type in the CATNETS scenario, which extend the `AgentSource` reasoner template. The `AgentSource` class is an implementation of the bilateral negotiation protocol which calls the Avalanche strategy for decision making. Additionally, it provides access to the evolutionary learning algorithm.

From a conceptual point of view, the agent calls the reasoner with a `Msg` object which contains all information according to the bidding language presented in the D1.1 deliverable and some additional identification information. The strategy interprets this object (`interpretMessage`) and returns a `Msg` object. This process separates the message propagation from the reasoning about the content using the defined negotiation protocol. The underlying infrastructure transports the content to its destination. The same concept is applied in the simulator and prototype environment.

The relevant attributes of the `Msg` class are in detail:

`MessageType`: The type of the negotiation message specifies the communicative act referred to the negotiation protocol. In the bilateral negotiation protocol of the catallactic reasoner the following values are used: cfp (call-for-proposal), accept, reject, proposal.

`ConversationID`: The identifier of the negotiation is an unique number generated for each new request during the creation of new cfp message. The values are random generated UUIDs of the Java built-in UUID generator.

`DocumentType`: This parameter signals the catallactic reasoner the type of price to reason about. A BID refers to the proposal type to be generated by a seller and an ASK relates to a buyer offer.

`ItemID`: This is the identifier of the traded good. In the current implementation, this could be any kind of text. In OptorSim, services usually are identified using their type (cs or bs for complex service or basic service) followed by a number. The bundles on the resource market are identified using a sequence of their single items. For example, ”cpu;mem;hdd” refers to a bundle of the three single bundle items cpu, memory (mem) and hard disk space (hdd).

`Price`: This is the current price of the traded good. On the service market, it is the price for a basic service instance; on the resource market it is the price for a resource bundle. In the current implementation, there is a predefined price for every resource bundle combination. The agent has a preference for trading certain resource bundles.

`Market`: The market parameter is used to assign the message to a market. In the CATNETS scenario, the values are: SERVICEMARKET, RESOURCEMARKET

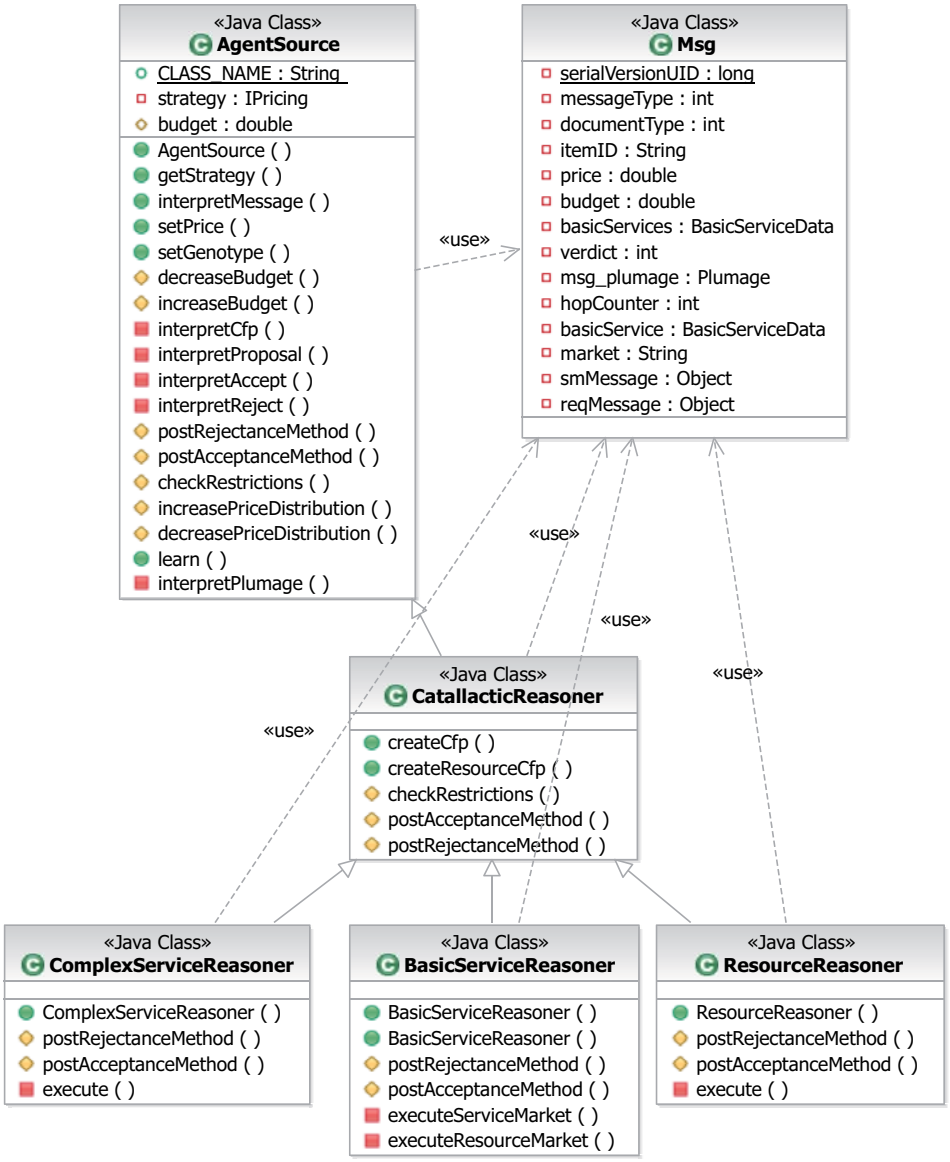

Figure 5.8: The interface of the catallactic reasoner.

`Verdict`: This is the verdict on bid set by the strategy. It relates to the result of the catallactic reasoner. Valid values are: accept, reject, proposal

`Plumage`: The plumage parameter represents a container for learning information. This container includes the fitness information and the genotype values of other agents which traded the same good.

The market agents use the described message object to decide the next operation. This operation is dependent on the message type of the market agent and its role in the market. Its general bidding behavior forming the two market is described in section 4.2. The implementation of the agents follows this description. The next section presents the initialization of strategy for the two markets in the CATNETS scenario.

initialization of the strategy for the service and resource market: The initialization of the reasoner splits into two areas: The first area is the initialization of the items and their initial price ranges, the second area is the initialization of the strategy behavior with a genotype. The parameters are:

Item Identifier (`itemID`): The product identifier to be traded. Both, the seller and the buyer have to initialize their starting price range specifying an minimum and maximum price. On the resource market, the product identifiers of all possible bundle combinations have to be initialized. This means for a bundle with 3 items, 7 item identifiers and their corresponding price ranges have to be initialized.

Initial price range (`minPrice`, `maxPrice`): The initial minimum and maximum price a buyer respectively a seller is willing to pay. The price range changes as described in formal description of the strategy (methods: increasePriceDistribution/decreasePriceDistribution).

Behavior of the strategy: The five parameters of the genotype (`satisfaction`, `acquisitiveness`, `priceNext`, `priceStep` and `weightMemory`) must also be initialised at the instantiation of the strategy. It is possible to initialize the strategy by using a random distribution or a pre-defined value set. The second option can be used to start simulation runs with a specific strategy. The behavior parameters change after every negotiation using the evolutionary learning algorithm (mutation and crossover operation).

Both markets are implemented as easy-to-use software modules providing an interface and its own configuration files. The same module is used in the OptorSim simulator and the middleware prototype.

5.2.2 Integration into OptorSim

This section briefly outlines the integration of the markets into OptorSim. For detailed description of how these concepts are implemented, the reader is referred to deliverable D2.2.

In OptorSim, the agents negotiate using a bilateral negotiation protocol. This negotiation protocol shows figure 5.9. The agents on both markets exchange their bids with the same bargaining protocol. They submit iterative proposals until they reach the end of the negotiation signaling an accept or reject.

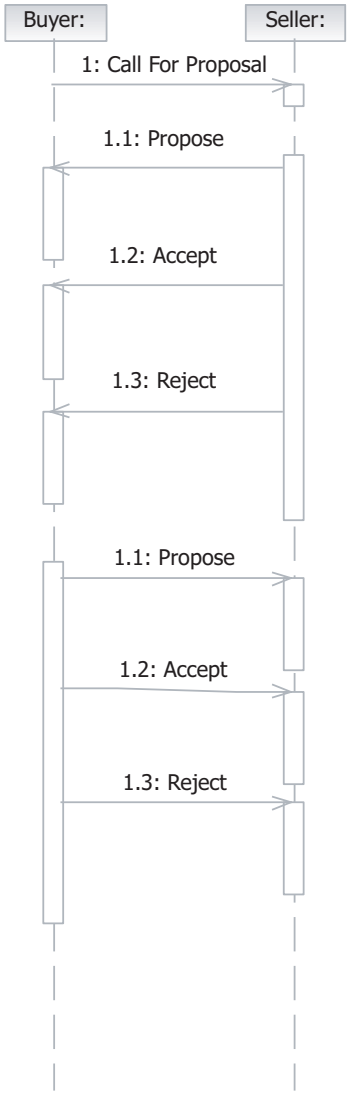

Figure 5.9: The bilateral negotiation protocol.

The communication between the trading agents is realized by means of messages.

Each time, a complex service agent wants to acquire of a basic service, it broadcasts a call-for-proposal message to other agents using its local Peer-to-Peer Mediator (see figure 5.10). This mediator provides access to the peer-to-peer infrastructure and is capable of routing messages to other agents. In case, a Peer-to-Peer Mediator receives a message, it forwards the message to its local agents. The call-for-proposal message is sent to local and remote basic service agents which are able to provide the service. They answer the call-for-proposal message by returning a proposal message to the sender. More communicative acts of the bargaining protocol are exchanged using point-to-point communication between the trading partners until they reach an agreement or cancel the negotiation.

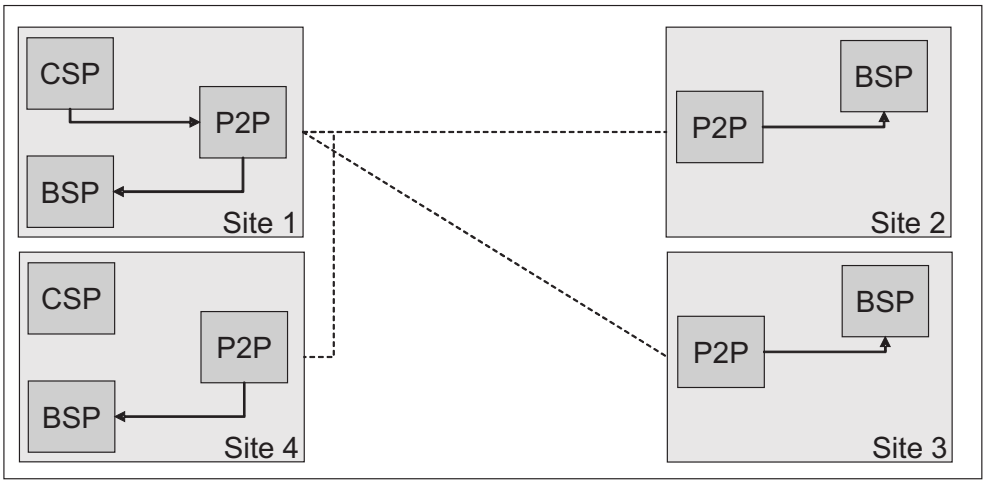

Figure 5.10: The broadcast of a Call-For-Proposal message using the OptorSim infrastructure.

After the end of a negotiation, the trading agents broadcast their own learning information to any other agent, which trades the same good, first. In a second step, they evaluate the collected plumages to adapt their own behavior. Messages for the learning process are implemented in OptorSim.

A complete list of all implemented messages for the catallactic mechanism realizing the discovery, the bilateral bargaining protocol, learning and synchronization of the markets presents deliverable D2.2.

5.2.3 Results from Simulation Runs

For a proof-of-concept evaluation of the simulator, this section presents preliminary results. The results are neither meant to be convincing nor to be statistically evident. Simulating large scenarios is the main effort of year three.

For the simulations, the rudiment scenario of Section 5.1.3 is used. The metrics negotiation time and allocation rate for two different behavior settings (genotypes) of the trading agents. The genotypes are:

Genotype 1: This genotype setting shows a strong cooperative behavior of the agents on the service market and on the resource market. On both market, the same genotype is used. The parameters are set to: `acquisitiveness` = 0.2, `priceStep` = 0.6, `priceNext` = 0.5, `satisfaction` = 0.9, `weightMemory` = 0.5.

Genotype 2: For the second genotype the same parameter setting as on the service market was applied. The genotype on the resource market exhibits a more profit oriented behavior. The trading partner are not willing to make frequent concessions. The genotype is on the resource market: `acquisitiveness` = 0.5, `priceStep` = 0.1, `priceNext` = 0.5, `satisfaction` = 0.9, `weightMemory` = 0.5.

Figure 5.11 and Figure 5.12 present the results for 10 simulation runs and both genotype settings. 50 requests per simulation run are submitted. The allocation rate clearly decreases for the second genotype, whereas the negotiation time stays quite the same for different genotypes. This is obvious: A more profit oriented behavior will lead to more negotiations rounds which let increase the influence of the time pressure represented by the `satisfaction` parameter. Using the same `satisfaction` value in both cases, the allocation rate declines.

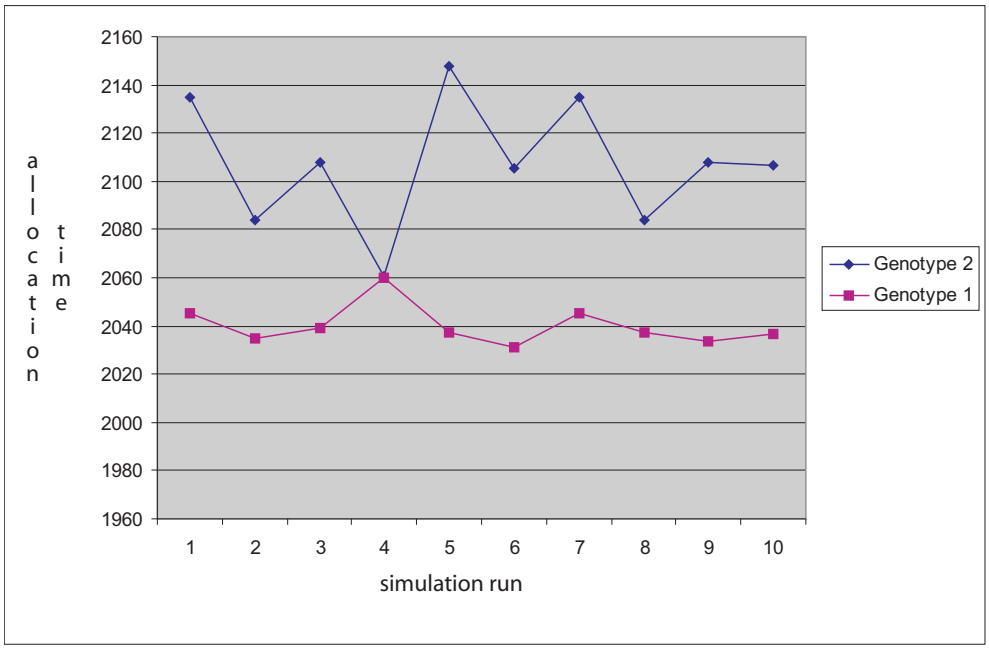

Figure 5.11: Negotiation times for 10 simulations runs and two different genotypes.

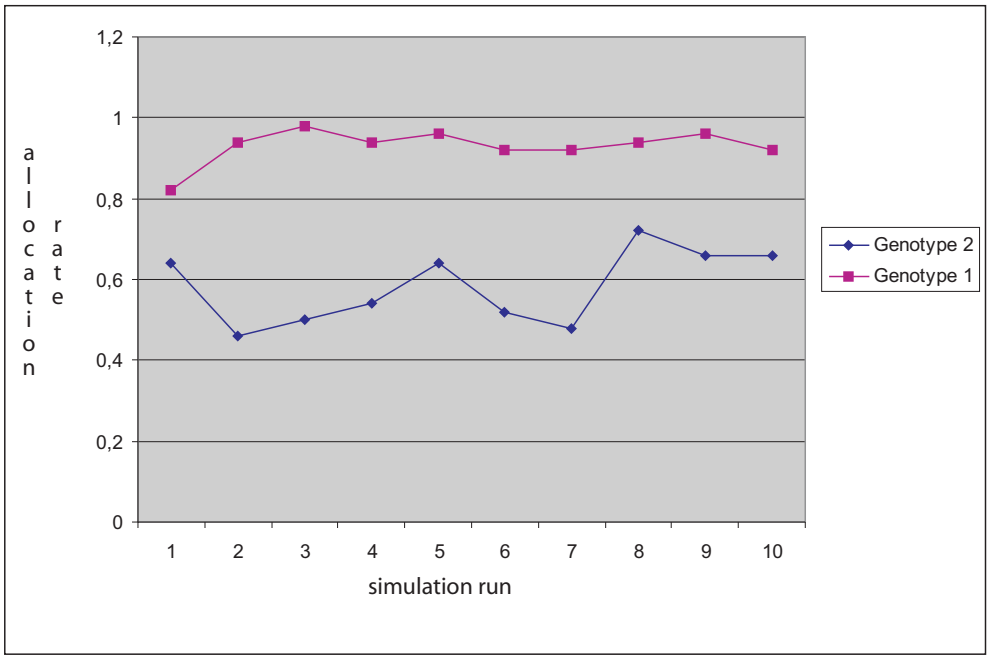

Figure 5.12: Allocation rates for 10 simulation runs and two different genotypes.

Chapter 6

Integration of Catallactic mechanism into middleware

The decentralized bargaining strategy is implemented in the middleware enabling the analysis of the strategy behavior in a real prototype environment. This chapter focuses on the integration of the Catallactic approach into the middleware. For the implementation of the core strategy algorithm, the reader is referred to deliverable D1.1. In this chapter, the interface of the strategy implementation, the software agents developed for the middleware, their interfaces and configuration files are described.

Section 6.1 present an in-depth introduction to the middleware environment used for the software agent development. This includes interfaces and configuration issues of the middleware developed in WP 3 and the most important parts of internal agent structure and its configuration. Additionally, this sections contains the specification of external interfaces for customized plug-ins enabling a flexible adaptation to different service types and resource bundles.

Section 6.2 presents preliminary results of one implemented scenario created together with WP 3. Interaction patterns of the prototype environment with the bargaining strategy integration in the middleware are examined. The results of the integration testing constitute further optimization issues of the strategy in year 3.

6.1 Description of the Integration of Decentralized Mechanisms into Middleware

The objective of this section is to describe the integration process. This comprehends a detailed look at the software agent API provided by the middleware and the structure of the developed agents including the integration of the catallactic strategy.

A general overview of the agents in the middleware shows figure 6.1. The agents are

used to perform the negotiations of services and resource bundles. Fulfilling their tasks, they use information from external components like the name of the service they sell or the current availability of resources from local resource managers.

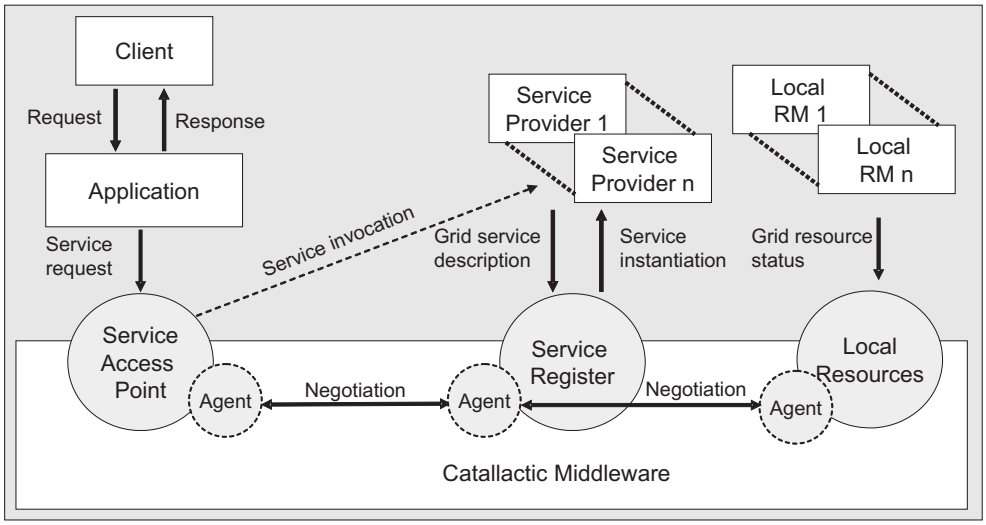

Figure 6.1: Overview of the catallactic middleware and their agents for negotiation

They receive a demand from an application using the Service Access Point as a mediator and deliver back the contracted service, which is immediately invoked and consumes the allocated resources.

6.1.1 Integration of the Agents Using the P2P Middleware Agent API

In the following, the implementation of the service market and the resource market is described using the P2P middleware agent API. Realizing these two markets of the CATNETS scenario, three different agent types have to be implemented: a complex service agent, a basic service agent and a resource agent. Every agent implementation extends the P2P agent API. Figure 6.2 shows the class diagram of the implemented agent templates. The API provides basic messaging capabilities for the agent communication like sender and receiver methods for point-to-point communication and group-casts. It is possible to schedule time-based events and to report the measured metrics. This functionality is used for the integration of the Catallactic strategy. Detailed information of all methods can be found in the documentation of the source code and in the deliverable of WP 3.

Beside the three agent types of the CATNETS scenario, a catallactic agent type is introduced, providing basic functionality for all three agent types:

Item group: This variable defines the group the agent belongs to. The item group is set at the initialisation of the agent. It relates to the items the agent is able to buy. The item group parameter is the filter for the broadcasts of the underlying p2p network.

Figure 6.2: The P2P agents of the middleware

Therefore, every agent has to join a group enabling him to receive call for proposal broadcasts and learning information from other agents which belong to the same item group.

Execution state: The execution state of an agent indicates its current status during the processing of incoming messages. An agent is not able to perform parallel negotiations as this will reduce complexity of the system. The current implementation uses 4 execution states: `IDLE`, `WAITING-FOR-PROPOSALS`, `BARGAINING` and `LEARNING`. Agent implementations with a buyer role (complex service and basic service) have the states and transition shown in figure 6.3 whereas agent implementations of a seller role (basic service and resource) act like depicted in picture 6.4.

After the initialisation of the agent, a buyer agent is in state `IDLE` waiting for a new demand or call for proposal messages. These are stored in their corresponding input queues and sequentially broadcasted to possible negotiation partners specified by the item group parameter. Also a corresponding discovery timeout event is scheduled. The buyer agent is now in the state `WAITING-FOR-PROPOSALS` where he collects proposal messages from seller agents. After reaching the timeout, the proposals are ranked and the cheapest offer is selected for further negotiation, using the bilateral bargaining protocol. Any other proposals are rejected. If the negotiation ended successfully, the agent changes its state to `LEARNING` processing any plumage messages of the plumage queue. Now, the agent handles the stored demand or call for proposal messages.

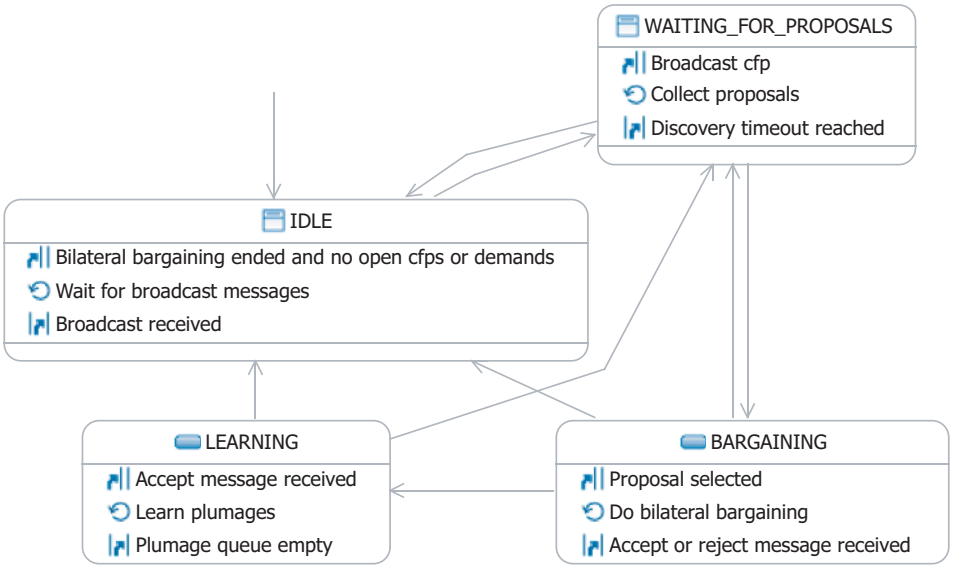

Figure 6.3: Execution state of a buyer agent

The states of a seller implementation are similar to the states of a buyer implemen-

tation described above. The buyer implementation reacts to call-for-proposal messages and answers them. The `WAITING-FOR-PROPOSALS` state is not needed here.

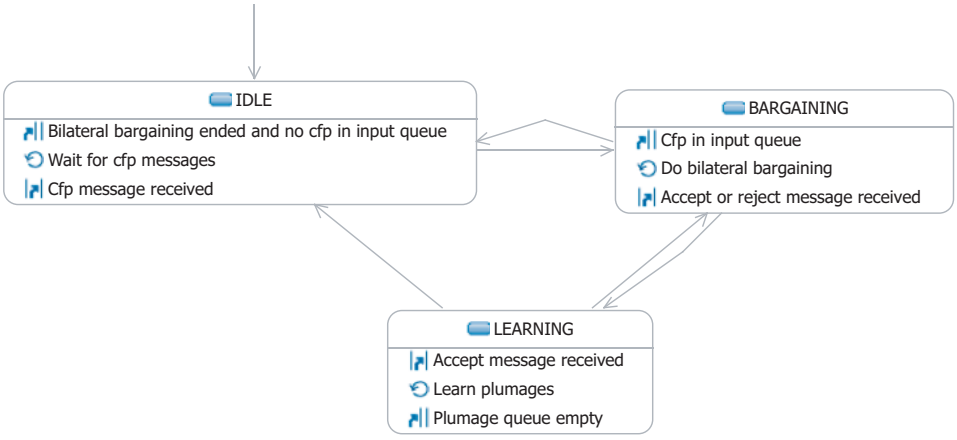

Figure 6.4: Execution state of a seller agent

Send message: This method replies a sender in a point-to-point communication. The recipient gets this messages through his handleMessage method of the `P2pAgent` class.

Broadcast message: The method broadcasts call-for-proposal and learning messages to the specified receiver group. The recipient gets this message through his `handleGroupCast` method.

An example messaging scenario shows figure 6.5. The Catallactic Access Point (CAP) broadcasts the received demand to the a complex service agent, which is able to handle this demand. The complex service agent notifies the basic service agents using a group cast. Both, the basic service agent and the complex service agent negotiate the price of the service using a bilateral negotiation protocol and making decision about the proposals with the avalanche reasoner classes. There, every middleware agent type has its related reasoner entity enabling customized strategy behavior. The middleware agents use the `interpretMessage` method of the strategy with the message content object to start the decision making process. The method returns the new content for the specified recipient. A confirm or cancel message signals the complex service if the resource negotiation was successful.

The negotiation on the resource market follows bidding strategy B (see section 4.2) synchronizing the two markets. First the agents negotiate on the service market and second, after a successful negotiation on the service market, the negotiation on the resource market takes place. At this point the middleware implementation differs from the simulator implementation. An external interface of the basic service implementation gives

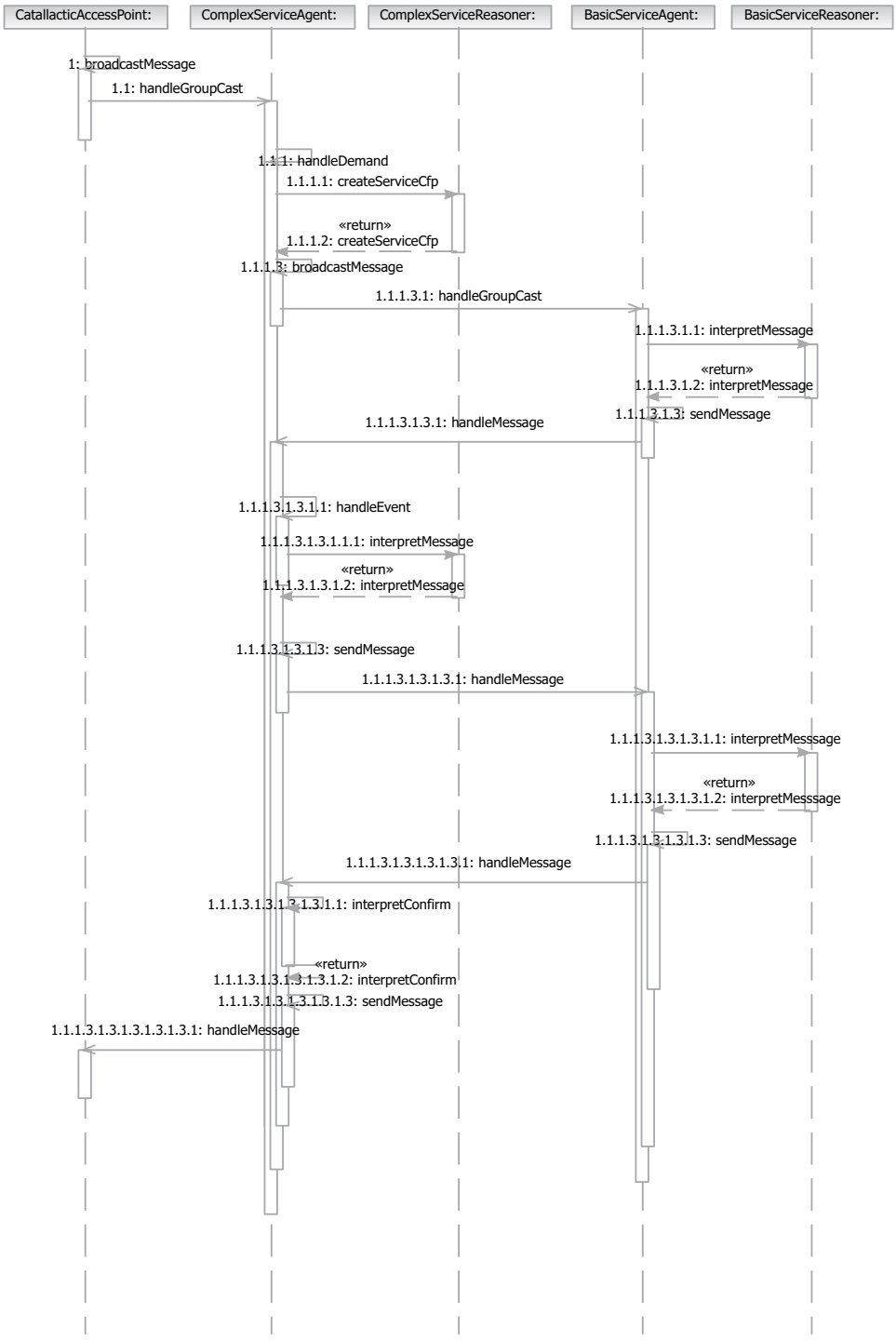

Figure 6.5: An example messaging sequence for a service market negotiation

the possibility to plug-in an external mapping table for the creation of a call-for-proposal message on the resource market (see figure 6.6). The amount a asked resources can differ for different basic service requests depends on attributes like the amount of data to be analyzed. Also the resource agents use an interface to external local resource managers determining the available resources to sell (see figure 6.6). It is assumed that each resource has its dedicated local resource manager. If an resource agent is able to sell a bundle of three resource items, three plug-ins for local resource managers have to be defined at the initialisation phase of the agent.

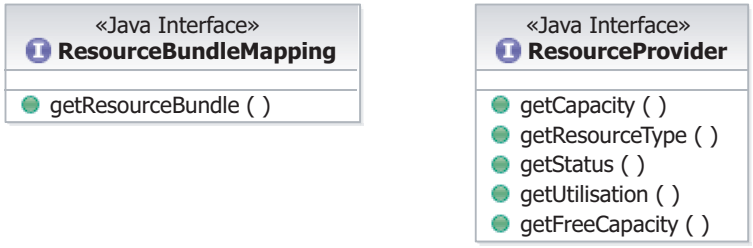

Figure 6.6: External interfaces of the middleware agents

The following section shows an implemented scenario with the middleware agents, which the prototype application uses for allocation of complex services. With this scenario, first results are presented.

6.2 Preliminary Results from Decentralized Mechanisms in Middleware Scenarios

The agents are integrated in the middleware as described in the previous section. For functional tests, a small scenario with few agents was created. Due to technical complexity, the integration in the prototype scenario using a real Grid system was postponed to the beginning of year three. Therefore, this section provides an insight about the functional test scenario.

For the scenario, several dummy agents simulating the local resource managers and the demands of the application prototype act as placeholders for the real environment. The focus of the tests lies on the correct behavior of the messaging of the agents, which is a complex part in the decentralized market mechanism. No deadlocks should appear in the distributed system.

The setup of the scenario looks as shown in figure 6.7. One application dummy agent submits requests in a loop to one complex service agent. Two basic service agents and two resource agents are available for fulfilling of the application demands.

The application demand contains a request for one basic service. A basic service uses the resource mapping model to create a request for a resource bundle on the resource mar-

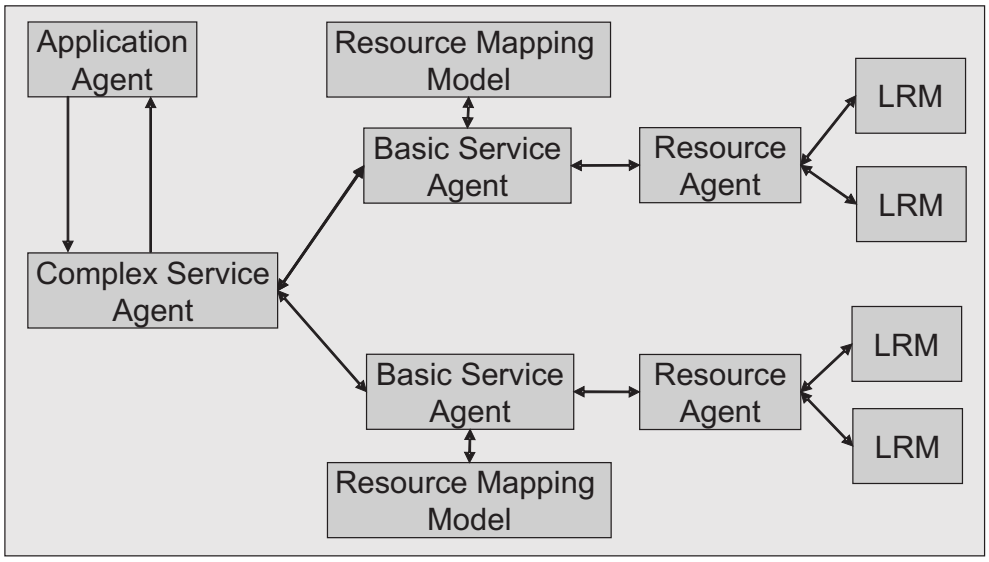

Figure 6.7: Simple test scenario for the middleware agents

ket. The request has to be delivered by one of the resources or both of them depending on their free capacity. Local resource manager dummies deliver the current status of the resource for allocation. They decrease the available resources after a successful negotiation and increase their available resources after an execution timeout occurs. The execution timeout was set to a fixed number.

In this scenario, two metrics are measured: the allocation rate (Figure 6.9) and the negotiation time (Figure 6.8). Two runs were executed with 20 requests per test run. The genotype was set to `acquisitiveness` = 0.2, `satisfaction` = 0.9, `priceStep` = 0.6, `priceNext` = 0.5, and `weightMemory` = 0.5.

Using this genotype, the agents tend to make concessions (`acquisitiveness` = 0.2) at a high rate. Additionally, the concession level is set to 60% of the trading range (`priceStep`). This leads to fast agreements.

The allocation rate in the first run was 90% and in the second run 85%. Comparing both diagrams, it seems that the outcome of an allocation does not influence the negotiation time.

Most time during the negotiation was spent waiting for the discovery timeout. The discovery timeout in the setting was set to 4000ms, 2000ms on the service market and 2000ms on the resource market. Blinding out the discovery time, the bilateral negotiation lasts between 800ms and 220ms.

For better conclusion about these results, more tests have to be conducted. At the current state, a more detailed analysis is not possible. This will be a major issue in year 3.

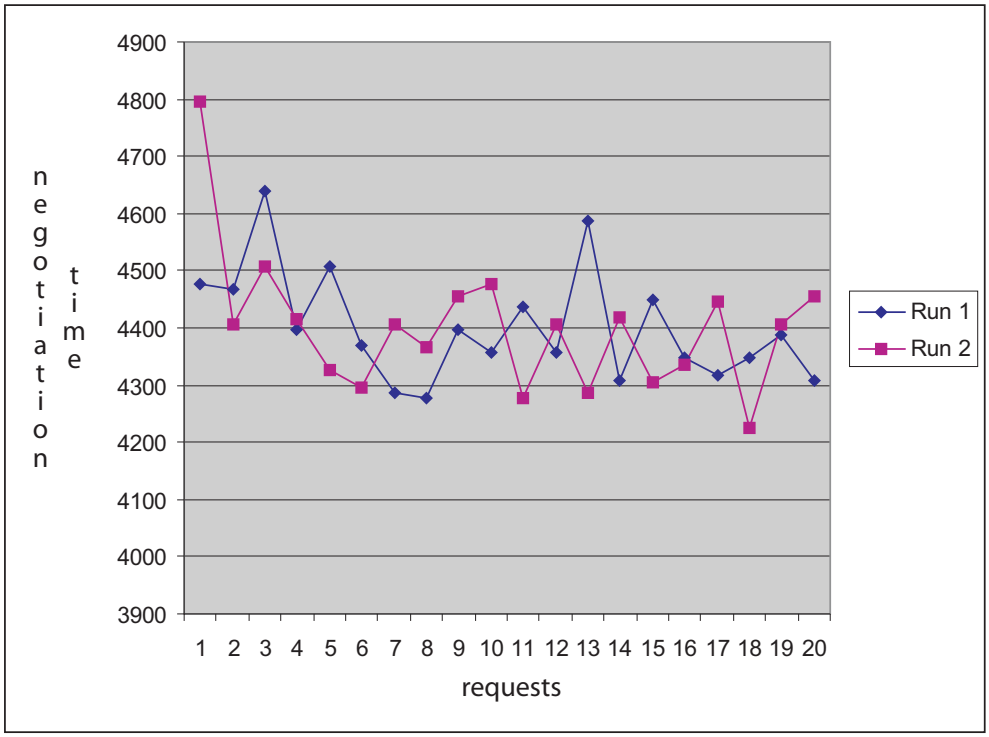

Figure 6.8: Negotiation time for two test runs

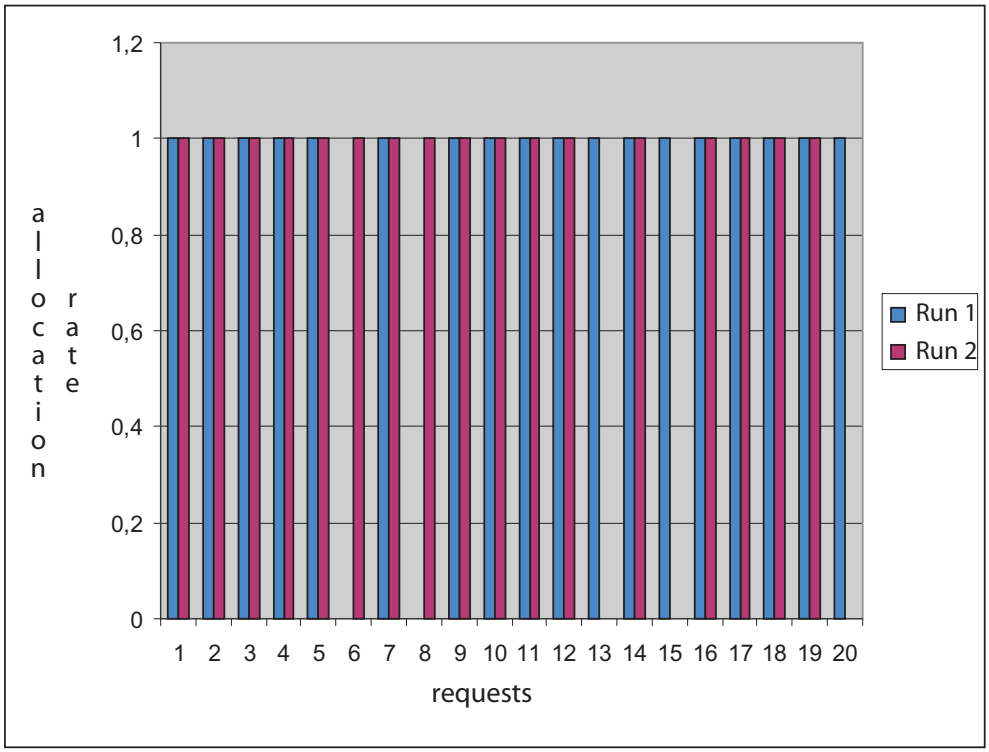

Figure 6.9: Allocation rate for two test runs

Chapter 7

Relations to other WPs

7.1 WP2

The objective of WP2 is to provide a simulator for ALNs. Our relations to WP2 are as follows:

- The centralized auctions and the decentralized bargaining strategies are integrated into the simulator. This includes a depth collaboration concerning the simulation model and the message system. Together with WP2, we developed the software patterns and components that are required to simulate market based ALNs.

7.2 WP3

The objective of WP3 is to have a proof-of-concept application that is used to demonstrate the ideas being developed, among others, in WP1. The relations to WP3 are as follows:

- The decentralized bargaining strategy is implemented in the middleware enabling the analysis of the strategy behavior in a real prototype environment. Scenarios for using the strategy were developed together with WP3. These scenarios describe interaction pattern of the prototype environment with the bargaining strategy integration in the middleware.
- The centralized cases (auctions) are not implemented in the prototype. We profit, however, from the knowledge provided by WP3: We see what can be realized in a price-based resource allocation mechanism, what is conceptually required for a successful application of markets in ALN's, and we learn which theoretical concepts are not implementable from an application point-of-view. For instance, we have designed a market language (i.e. how agents formulate their bids) by means of WS-Agreement to ensure compatibility with the prototype and common GGF standards.

7.3 WP4

The goal of WP4 is to evaluate the performance of the Catallactic approach by means of a simulator (see deliverable year 2 of WP2) and a prototype (see deliverable year 2 of WP3). The relations to WP4 are as follows:

- The identification and formalization of relevant metrics to compare centralized and decentralized market mechanisms.
- Collaboration with the development of the performance measuring framework, due to the fact that some of the measured metrics are taken at the economic agent level. Reporting of measured data to the performance measuring framework.

Chapter 8

Outlook

In this chapter the achievements from workpackage 1 during Y2 of the project are summarized and an outlook on the work that will be content to the next 12 months is given. Section 8.1 focuses on the work that has been performed in the last 12 months. A brief review is given on how this relates to the first project year. In Section 8.2 an outlook on the work that will be performed in year 3 is given – also under consideration of the DOW.

8.1 Review

As described in the introduction, the main contribution of the CATNETS project is the comparison of centralized and decentralized economic allocation mechanisms for resources in Grids and application layer networks (ALNs). In order to achieve this, the project has been divided into five workpackages. The content of the individual workpackage as well as the division and integration of those has been elaborated on in Chapter 1 and Chapter 7.

In this report, the line of work carried out in the second project year (month 13 to month 24) is described. Thereby, Chapter 3 focuses on the formal description of the centralized and decentralized mechanisms. The centralized mechanisms, which also have been content to year 1 report, are briefly discussed and an in depth presentation of the decentralized mechanisms as well as the learning algorithms and negotiation strategies is given. The notion of two different markets – a resource and a service market – is introduced. Both markets are interconnected via a intermediaries.

After this, Chapter 4 clarifies *when* and *what* a participant in resource and service markets bids. The definitions provided here are substantial for the design of both (i) the implementation of the mechanisms in the simulator as well as (ii) the integration of the decentralized mechanisms into the middleware.

In Chapter 5 the integration of the centralized and decentralized mechanisms into

the OptorSim simulator framework is described. Besides the preparatory work that is content to the chapters before, this has been the most demanding and extensive work that has been carried out in workpackage 1 in year 2. The challenge here is to provide a flexible, adaptable and dynamic simulation framework that enables both, simulations based on centralized and decentralized market setups in one scenario in order to keep the results comparable. Additionally to the issues concerning the implementation itself, preliminary simulation results from both, the centralized and the decentralized case are presented. These results are, of course, some proof-of-concept outlines that simply show, that the proposed implementations work. The task to provide measurable and statistically relevant quantitative data and results will be content to project year 3.

Chapter 6 focuses on the description of the integration of the decentralized – Catallactic – mechanisms into the Grid/ALN middleware. Next to the technical integration and the configuration described also in workpackage 3 report, preliminary results of a created scenario are described. The work elaborated on in this chapter constitutes the second important – and time consuming – column that has been content to workpackage 1 in project year 2. Further refinements, specification of more detailed scenarios as well as creation of quantitative results from the scenarios will be content to project year 3.

8.2 Content to Y3

After the integration of the centralized and the decentralized economic mechanisms into the simulator and the integration of the centralized mechanism into the middleware is finished and refined the following tasks will be the main issues for workpackage 1:

- *2.3 Simulation of application layer networks and refinement:* Here, the efforts in workpackage 1 will be focused on the calibration, validation and verification of the simulation model. Additional issues will be the assistance of workpackage 2 leaders in carrying our simulations, obtaining and evaluating large-scale data from simulation runs as well as to interpret the results derived from the applied metrics.

- *3.3 Performance measuring components for experiments:* Concerning this issue, most work is carried out in workpackage 3. However, the economic expertise in designing and monitoring the performance evaluation both, from a technical and an economic perspective will be our mission for year 3.

- *3.4 Distributed application to execute on economic-enhanced Grid/P2P platform and middleware integration:* In this task, the integration of middleware concepts for novel approaches in Grid and P2P architectures will be evaluated. Besides generating substantial results, the target of workpackage 1 will be her to assist workpackage 3 leaders in provisioning of a strong footprint of the work performed in CATNETS in the Grid/SOA/P2P/Pervasive Computing communities.

- *4.4 Performance analysis, comparison, evaluation:* As all other participants, it will be one of the core issues to analyze, document and compare the evaluation of the proposed mechanisms. Workpackage 1 will contribute to this effort by assisting workpackage 4 protagonists in order to show the efficiency and effectiveness of the proposed mechanisms in appropriate scenarios and find channels to distribute these results into all relevant communities.

Bibliography

[AG04] Nadia Ben Azzouna and Fabrice Guillemin. Charcteristic of ip traffic in commercial wide area networks. In *Proceedings of the International Conference on Computing, Communcations and Control Technologies (CCCT'2004), Austin, Texas (TX)*, August 2004.

[AH00] E. Adar and B.A Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[Bre02] T. Brenner. A behavioural learning approach to the dynamics of prices. *Computational Economics*, pages 67–94, 2002.

[Car04] Nicholas G. Carr. *Does IT Matter? Information Technology and the Corrosion of Competitive Advantage*. Harvard Business School Press, May 2004.

[CJSF06] P. Chacin, L. Joita, B. Schnizler, and F. Freitag. Flexible architecture for supporting auctions in grids. In *Proceedings of the 2nd International Workshop On Smart Grid Technologies 2006 (SGT2006) Workshop*, 2006.

[CSSZ06] Gaetano Calabrese, Björn Schnizler, Werner Streitberger, and Floriano Zini. D2.2: Annual report of wp2. Catnets deliverable, ITC-irst Trento, University of Karlsruhe, University of Bayreuth, 2006.

[ESMP03] T. Eymann, S. Sackmann, G. Müller, and I. Pippow. Hayek's catallaxy: A forward-looking concept for information systems. In *Proceedings of the American Conference on Information Systems (AMCIS), Tampa, Florida*, 2003.

[ESP98] Torsten Eymann, Detlef Schoder, and Boris Padovan. Avalanche - an agent based value chain coordination experiment. In *Workshop on Artificial Societies and Computational Markets (ASCMA'98)*, pages 48–53, Minneapolis, 1998.

[Eym01] Torsten Eymann. Decentralized economic coordination in multi-agent systems. In Hans-Ulrich Buhl, F. Huther, and A. Reitwiesner, editors, *Information Age Economy. Proceedings WI-2001.*, pages 575–588, Heidelberg, 2001. Physica Verlag.

[Fri91] D. Friedman. The double auction market institution: A survey. In D. Friedman and J. Rust, editors, *The Double Auction Market - Institutions, Theories, and Evidence*, pages 3–26. Cambridge MA, Perseus Publishing, 1991.

[HBKC89] F.A.v. Hayek, W.W. Bartley, P.G. Klein, and B. Caldwell. The collected works of f.a. hayek. *University of Chicago Press*, 1989.

[KC03] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

[KR95] J.H. Kagel and A.E. Roth. *The handbook of experimental economics*. Princeton University Press, 1995.

[Mat99] Humberto R. Maturana. The organization of the living: a theory of the living organization. *Int. J. Hum.-Comput. Stud.*, 51(2):149–168, 1999.

[PKE01] David C. Parkes, Jayant Kalagnanam, and Marta Eso. Achieving budget-balance with vickrey-based payment schemes in exchanges. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[Pru81] D.G. Pruitt. Negotiation behavior. *Organizational and occupational psychology. New York: Academic Press*, 1981.

[PT02] W. H. Press and S. A. Teukolsky. *Numerical Recipes in C++ - The Art of Scientific Computing*. Cambridge, MA, Cambridge University Press, 2002.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. Technical report, Berkeley Press, 2001.

[Ros94] G. Rosenschein, J. S.; Zlotkin. Rules of encounter - designing conventions for automated negotiation among computers. *MIT Press, Cambridge*, 1994.

[Smi62] V.L. Smith. An experimental study of competitive market behavior. *Journal of Political Economy*, 70:111–137, 1962.

[SNV+05a] Björn Schnizler, Dirk Neumann, Daniel Veit, Mauro Napoletano, Michele Catalano, Mauro Gallegati, Michael Reinicke, Werner Streitberger, and Torsten Eymann. f: Environmental analysis for application layer networks. Catnets deliverable, University of Karlsruhe, Universitá delle Marche Ancona, University of Bayreuth, 2005.

[SNV+05b] Björn Schnizler, Dirk Neumann, Daniel Veit, Michael Reinicke, Werner Streitberger, Torsten Eymann, Felix Freitag, Isaac Chao, and Pablo Chacin. Deliverable 1.1; wp 1: Theoretical and computational basis. Technical report, CATNETS, 2005.

[SNVW06] Björn Schnizler, Dirk Neumann, Daniel Veit, and Christof Weinhardt. Trading Grid Services – A Multi-attribute Combinatorial Approach. *European Journal of Operational Research, forthcoming*, 2006.

[Tes97] L. Tesfatsion. How economists can get alife. In *The Economy as a Evolving Complex System II*, pages 533–564. Arthur, W.B. and Durlauf, S. and Lane, D.A. (Hrsg.), 1997.

[Var94] Hal R. Varian. *Mikrokonomie*. Oldenbourg, 1994.

[Wei99] Gerhard Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[Wie98] N. Wiener. The history and prehistory of cybernetics. *Kybernetes*, 27:29–37, 1998.